NASA CR-
147878

TECHNOLOGY

RESEARCH

and

DEVELOPMENT

INCORPORATED

LIFE SCIENCES DIVISION

SPECIAL REPORT

EMGAN: A Computer Program for Time and Frequency Domain Reduction of Electromyographic Data

September 5, 1975

CONTRACT NAS 9-13291

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas 77058

17311 EL CAMINO REAL • HOUSTON, TEXAS 77058

TECHNOLOGY INCORPORATED
LIFE SCIENCES DIVISION
HOUSTON, TEXAS

SPECIAL REPORT

EMGAN: A Computer Program for Time and Frequency Domain Reduction of Electromyographic Data

Septebmer 5, 1975

SPECIALIZED CARDIOVASCULAR STUDIES

CONTRACT NAS 9-13291

PREPARED BY:

William N. Hursta

William N. Hursta
Research Bioengineer

APPROVED BY:

Joseph T. Baker

Joseph T. Baker
Project Leader

APPROVED BY:

[illegible]

T. Wayne Holt, Manager
Life Sciences Division

SPECIAL REPORT

EMGAN: A Computer Program for Time and Frequency Domain Reduction of Electromyographic Data

September 5, 1975

CONTRACT NAS 9-13291

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas 77058

TABLE OF CONTENTS

INTRODUCTION

An experiment in Electromyography (EMG) utilizing surface electrode techniques was developed for the Apollo-Soyuz Test Project. The objective of the study was to obtain quantitative measure of changes in muscle characteristics as a result of exposure to weightlessness (see Appendix A). This report describes the computer program, EMGAN, which was written to provide first order data reduction for the experiment.

EMG signals are produced by the membrane depolarization of muscle fibers during a muscle contraction. Surface electrodes detect a spatially summated signal from a large number of muscle fibers commonly called an interference pattern. An interference pattern is usually so complex that analysis through signal morphology is extremely difficult if not impossible. However, because of the ease of application and noninvasive character of surface electrodes and the often desirable feature of studying large groups of muscle fibers, methods have been sought to evaluate the surface interference pattern.

In the time domain a valuable technique to characterize an interference pattern has been the root-mean-square (RMS) value of the signal. A similar technique has been to integrate the interference pattern over a given time length. A linear relationship is usually found between the integrated EMG or the RMS value and the force exerted by the muscle up to some maximum.

More recently with the increased access to the digital computers and the availability of an algorithm to economically calculate discrete Fourier transforms, it has become common to process EMG interference patterns in the frequency domain. Muscle fatigue and certain myopathic conditions can be recognized through changes in muscle frequency spectra.

DATA PREPARATION

The protocol for the ASTP Myography required that two types of data be recorded, digitized, and processed: the force being exerted by a muscle group and the surface EMG signal. The data was originally recorded on a 4-channel Tandberg analog recorder with one-channel of IRIG B time code, one-channel of force data, and two-channels of simultaneous EMG signal (two different muscles).

A digital tape was prepared from the analog data using the General A/D program (see Appendix H) available on the CF-16A mini-computer located in the Sigma 3 Computer Room. Analog force data was low pass filtered at 10 Hz before being digitized at 20 samples/second. Analog EMG data was low pass filtered at 400 Hz. Consistent with this frequency limit a digitizing rate of 1000 samples/second was chosen as the minimum necessary for power spectral density analysis [2].

The digitized data was organized into half second records with 535 words per record. The first twenty-five words were header information with ten words of digitized force data interspaced into 500 words of EMG data (see data format in Appendix B). Only one-channel of EMG data was digitized with its associated force data at any one time. Although not required by EMGAN, a data tape was usually constructed with the first file containing force calibration and subsequent files containing EMG data from each muscle with the appropriate EMG calibration signals.

PROGRAM CHARACTERISTICS

EMGAN exists in two versions. One version was written to execute on the Biomedical Sciences Division Xerox Sigma 3 Computer using the Computek 400/20 CRT terminal for graphical output. A second version was written to execute on the Cardiovascular Laboratory's Data General Nova 820 mini-computer. Both programs are essentially identical to the user. However, there are a number of programming differences made necessary by differences in Xerox and Data General's Fortran IV, supporting software packages (graphical and tape I/0), and peripheral hardware. Whenever these differences are sufficiently important in describing the program they will be noted. Complete program listings of both versions with their associated load maps are given in the Appendices F and G.

An EMG system tape exists for the Sigma 3 Computer which contains an appropriately sized operating system. EMGAN exists in absolute binary as a file in the User Processor area of the Rad. A source listing of EMGAN exists both on magnetic tape stored in the Sigma 3 Computer Room and on punched cards in the Cardiovascular Laboratory. Both source listings and absolute binary files of the Nova version of EMGAN are stored on disk pack and on magnetic tape in the Cardiovascular Laboratory. The absolute binary files are named EMGAN.SV and EMGAN.OL (both are needed for program execution). The source listing is located in separate files for the main program and each subroutine with each file named for the subroutine it contains. How to execute a run with either computer version is explained in a later section. The computer peripherals required to fully execute EMGAN are a card reader, line printer, two 9-track tape drives, and the Computek terminal (Sigma) or the Tektronix 4014 terminal (Nova).

EMGAN's execution is totally controlled by the data card information received. Force and EMG data slices are processed with separate data cards. The maximum length of a force data slice is 120 seconds with previous force calibration required. After the raw data is converted into pounds it is averaged over a user selected interval for output to the line printer. The maximum length of an EMG data slice is four seconds with previous EMG calibration required. After scaling into microvolts, an integrated value is found for the EMG data slice. The discrete Fourier transform of the data is taken. The raw power spectral density (PSD) of the data up to 400 Hz is calculated and smoothed over a frequency bandwidth selected by the user. The raw PSD with associated header information is outputed to magnetic tape for use in any subsequent processing program. The smoothed PSD is outputed to the line printer along with accessory information such as a normalized PSD, percent contribution of each bandwidth to the whole spectrum, the mean and standard deviation of the spectrum, and time domain integration and mean square value of the data slice.

EMGAN run time is variable and depends on the number and kind of data cards, the length of the EMG data slices, whether or not the 60 Hz filter is used, data location on the input tape (processing data in sequential order on the tape is much faster than skipping back and forth on the input tape), the number of data plots, and whether execution is on the Sigma or Nova. For the ASTP EMG experiments a typical run consisting of calibration cards, 16 force data slice cards, and 28 4-second EMG data slice cards (no filtering or plotting) required approximately 25 minutes on the Sigma and 35 minutes on the Nova.

EMGAN DESCRIPTION

In describing the various subroutines, differences in the Nova version from the Sigma version will be set off with a ***. A table of formulas used in the program is listed in Appendix D.

Main Program:

The Main Program's basic task is to load the needed overlays in the proper sequence. A flowchart is given in Appendix C. The program is organized into a large do loop with one pass through the loop for each data card to be processed. A check is made to determine if calibration data has been received before the respective data type, force or EMG, can be processed. If the user should request the processing of data without calibration, the program outputs an error message and halts.

*** The magnetic tape units are initialized and the output tape file is opened. Upon completion of the program the tape units are released.

Subroutine CINPUT

Subroutine CINPUT inputs all data card information. Data card format is given in Appendix B. As each card is read it is outputed to the line printer to aid the user in verification. If the number of data cards to be processed is read to be greater than 200, an appropriate error message is given and the program halts. All data card information is stored in a scratch file, ICARD, in the user data area of the RAD. As each data card is to be processed ICARD is read back into main memory. The program variables are then set to the corresponding data card values.

*** The scratch file, ICARD, is created on the disk pack by the Nova program if it does not already exist. For the Sigma version the user should create ICARD (1 record, 6000 bytes, random format) if it does not already exist before attempting to execute EMGAN.

Subroutine TINPUT

Subroutine TINPUT acquires the digitized data from the input tape. Tape files are numbered from 00 to 99. The routine keeps track of the current file being accessed. If a data card requests another file, TINPUT skips the tape either forward or backward to the beginning of the requested file. Subroutine FIND is called to position the tape at the start of the data slice within the file. TINPUT reads either force or EMG data according to data card request. After the data has been read, the tape is backed up to the beginning of the data slice.

If an EOT is encountered during a search for a requested file, an error message is given and the program halts. If an EOF or EOT is encountered during the input of a data slice, a error message is given and the program halts.

*** Nova Fortran calls for magnetic tape I/0 are considerably different from the QINOUT package used on the Sigma. Because Nova error checking is more comprehensive, error messages given by EMGAN resulting from tape I/0 problems contain much more information.

Subroutine FIND

Subroutine FIND searches the currently accessed tape file for the start time of the data slice. If an EOF is encountered before the start time is found, the tape is rewound to the beginning of the file. A second search is then made through the entire file. If the start time is still not found, an error message is given and the program halts.

Subroutine OUT60

Subroutine OUT60 is a digital notch filter which can be called to remove 60 Hz and its odd harmonics from EMG data. If at all possible filtering should be avoided, since no distinction can be made between 60 Hz present in the true data and that which may have been picked up from extraneous sources. Since a large number of sines and cosines are generated, considerable time is added to the processing of data slices.

Subroutine DCAL

Subroutine DCAL processes force and EMG calibration data to obtain the scale factors needed to change the data from raw counts to engineering units. After the scale factors have been calculated, the calibration is converted to engineering units for plotting.

A force calibration data slice consists of two DC levels, the first level corresponding to zero pounds of force, and a second level corresponding to a known amount of force (98 pounds in the ASTP experiment). The data slice is searched for a jump in the data to indicate the beginning of the second level. When the jump is found, the average number of counts three seconds before the jump is subtracted from the average number of counts three seconds after the jump. The difference is divided by the number of pounds represented by the jump to obtain the scale factor. If the jump is not detected, an error message is outputed and the program halts.

The EMG scale factor is calculated using the fact that for a zero mean, stationary signal the standard deviation of the signal is equal to its RMS value. (For the ASTP experiment a nominal 20 Hz, 350 microvolt RMS signal was used for EMG calibration). The calibration data first has any DC mean subtracted from it and the standard deviation of the data in counts is found. The scale factor is computed by dividing the standard deviation by the RMS magnitude of the calibration signal.

Subroutine GRAPH

Subroutine GRAPH plots on the Computek terminal, if requested by the data card, force and EMG calibration data, force data, and EMG data both in the time and frequency domain. Scaling for force and EMG data is variable according to the maximum amplitude of the data. Force data is plotted without smoothing (such as is done for the printed output). When time domain EMG data is plotted, only 1000 evenly spaced points are plotted no matter what the length of the data slice. The PSD plotted is smoothed and normalized to the maximum value of the spectrum. The return key must be struck for the program to continue whenever the terminal's bell is rung.

***As of the writing of this report the graphics package for the Tektronix 4014 terminal is not available. No plotting capability presently exists when executing EMGAN on the Nova. However, if plots are requested when executing on the Nova, no run error occurs.

Subroutine MODLINE, WORDS, and BELL

These routines refer to the Computek terminal and respectively, plot data arrays, output alphanumeric characters, and ring the Computek bell.

Subroutine EMG

Subroutine EMG ubtracts any DC offset from the unscaled EMG data. The data is then scaled into engineering units and the largest absolute value of the data array is found for use in plotting of the data. Simpson's rule is used to find the integrated value of the data in units of microvolt-seconds. The mean square value of data is calculated.

Subroutine FORCE

Subroutine FORCE scales the force data into engineering units. Compensation is made for force signals recorded at a higher gain than the force calibration. The first second of force data is assumed to be the zero force level. The user should insure that this is so for correct results. The maximum force in the data slice is found for use in plotting the data. Force data is then averaged over intervals requested by the data card for output to the line printer.

Subroutine FFTPSD

Subroutine FFTPSD calculates the raw power spectral density of an EMG data slice up to 400 Hz. It is suggested that the user read (2) for information on data analysis using PSD techniques and (1) for an excellent explanation of discrete Fourier transforms and the fast Fourier transform algorithm. A cosine taper is applied to the first and last tenths of the data slice to reduce leakage. The Fourier transform of the data is then taken and the raw power spectral density is computed. The PSD amplitudes are corrected for reduction caused by the cosine taper. The raw PSD is outputed to magnetic tape for use in subsequent processing programs. The PSD is smoothed over the frequency bandwidth requested by the data card and the maximum PSD value found. A second PSD array is computed, normalized to the maximum PSD value. The area under the PSD is found and used to calculate the percentage each bandwidth contains of the total power and the cumulative of percentage of total power with increasing frequency. The expected value and variance of the PSD are computed.

Viewed statistically the raw PSD has very poor reliability. Each calculated value is an inconsistent estimate of the true value and has a possible random error of 100%. Smoothing the PSD can greatly reduce the random error and provide a better estimate over the smoothed bandwidth (2). The normalized standard error is calculated to provide information on the amount of random error associated with each PSD value.

Subroutine WINDOW

Subroutine WINDOW applies a cosine taper to the first and last tenths of the data slice to reduce leakage in the discrete Fourier transform.

Subroutines RFORT and FORT

Subroutine RFORT and FORT take the Fourier transform of the EMG data. FORT contains the actual fast Fourier transform algorithm and assumes a transform of N complex data points. RFORT compensates for the actual case of a transform of 2N real data points. The FORT computes an initial sine/cosine array and refers to this array when computing the transform. Repetitive computation of identical sines and cosines is thereby avoided, substantially increasing the speed of the transform at the cost of an extra array of size N/4. Both of these subroutines were received from Mr. Jack McBryde of Lockheed. The only modification was to not divide each data value by the number of data points (an alternate definition of the discrete Fourier transform). EMGAN can transform 1, 2, or 4 seconds of EMG data which corresponds to 1024, 2048, and 4096 data points, respectively. Computation time for the Fourier transform algorithm are as follows:

# of Points	Sigma Time	Nova Time
1024	6 seconds	10 seconds
2048	13 seconds	22 seconds
4096	29 seconds	47 seconds

*** Subroutines RFORT and FORT are named RFFT and FFT to avoid confusion with Nova Command Line Interpreter calls.

Subroutine PSDSAVE

Subroutine PSDSAVE outputs to magnetic tape the raw PSD of each EMG data slice. The PSD values are preceded by a 25-word header record. The format of the header record is located in Appendix B. The PSD of a one-second slice of data has 512 real values, a two-second slice has 1024 real values, and a four-second slice has 2048 real values. A PSD record contains 2048 integer words; therefore, a one-second PSD fills 1/2 of one record, a two-second PSD fills one full record, and a four-second slice fills two records. Two E0F's are written on the tape after the raw PSD is output and the tape then backs up to just before the

EOF's. At the end of an EMGAN run the output tape consists of a file containing all of raw PSD's from the run terminated by two EOF's.

Subroutine PRINT

Subroutine PRINT outputs to the line printer all of the processed force and EMG data. A page of header and calibration information is printed whenever the print switch on a calibration data card is set. The length of a force or PSD output is checked and if sufficiently long a two column list of the processed data is printed instead of a one column list. The frequency corresponding to each PSD value is the center of the bandwidth.

*** Subroutine PRINT is named POUT to avoid confusion with Nova Command Line Interpreter calls.

SAMPLE EMGAN RUN

To execute an EMGAN run on the Sigma 3 first load the EMG system tape. Place the data cards in the card reader and power up the Computek terminal. Mount the input data tape on unit 0 and the PSD output on unit 1. Assign the user input device to the teletype and type in "!EMGAN".

To execute an EMGAN run on the Nova load the EMG disk pack and bring up the system. Place the data cards in the card reader. Mount the input data tape on unit 0 and the output tape on unit 1. Type in "EMGAN". A message will be outputed to the terminal, "LOAD $CDR, STRIKE ANY KEY", upon entering any keyboard character, the data cards will be read and the program executed.

An example of an EMGAN input card deck with the printed and plotted output is in Appendix E.

PROGRAM IMPROVEMENTS

There are several additions that would be desirable in future work with EMGAN. The major addition would be a subroutine to check the EMG data slice prior to transform for stationarity and normal distribution. Suggestions for the necessary statistical techniques are found in (2).

An alternative FFT algorithm might be used which calculates sines and cosines as needed rather than establishing a table. The core storage saved would allow an 8196 real data point transform to be performed on the Sigma and, with proper sizing of operating system, on the Nova. However a price would be paid in increased processing time considerably beyond twice the transform time of 4096 real data point array.

Of a more minor nature subroutine GRAPH could be modified to output the muscle name with the EMG and PSD plots to allow easier later identification. Although force and EMG data are now processed separately, it could be useful to obtain the average force value over the period of the EMG data slice. The force value could be output with the EMG's PSD for more convenient association between the two.

REFERENCES

1. Brigham, E. O. The Fast Fourier Transform, Prentice-Hall Inc., Englewood Cliffs, N. J., 1974.

2. Bendat, J. S. and Piersol, A. G. Random Data: Analysis and Measurement Procedures, John Wiley and Sons, New York, N. Y., 1971

APPENDIX A

ASTP MYOGRAPHY DESCRIPTION

1.0 General Objective

The general objective of the experiment is to continue the study efforts begun in the Skylab program to identify and describe antigravity muscle dysfunction characteristics and consequences resulting from spaceflight.

2.0 Specific Objective

The purpose of the experiment is to assess changes that occur following a period of disuse. Some studies suggest that the first few days of exposure to 0-g may be significant to the ultimate muscle deconditioning resulting from longer missions. The relationship between muscle capability, in terms of strength or tension, fatigability, and muscle electrical activity will be investigated, as well as the differential effects of spaceflight disuse on "fast" and "slow" muscles.

3.0 Significance

Muscle function and condition may well be a critical determinant of man's capacity to endure the effects of long duration weightlessness as well as the readaptation to the earth's gravity, or to the gravity of other planets. Data collected in the experiment will aid in quantifying the muscle deconditioning which results from weightlessness.

In addition, an important spinoff will be the extension of an already considerable ground-based body of knowledge about the characteristics and consequences of muscle disuse. Heretofore, research in this area has relied on the "contrived" methods of surgical section and limb or torso fixation to produce the disuse effects. Spaceflight uniquely provides a "pure" form of neuromuscular system disuse.

4.0 Method

A. Concept

The measurement of muscle electrical activity is known as electromyography (EMG). The state of muscle function can be described by EMG measurements combined with knowledge of the force being exerted by the muscle.

Electromyographic studies have shown that skeletal muscle undergo changes in capability and composition when subjected to periods of disuse, i.e., periods of time when the contractile mechanisms are not subjected to the stresses and forces normally encountered. Also changes in biochemical constituents, as seen in

Skylab, such as calcium, potassium, etc., and alterations in enzymic constiuents such as ATP, and acetylcholinesterase have been shown, by ground studies to affect normal muscle function.

The EMG results of Skylab 3 skeletal muscle assessment provide ample evidence that normal muscle function is significantly altered by periods of weightlessness (disuse) of 56 days or more.

To investigate the effects of a shorter period of weightlessness on muscle function, a standardized test protocol and a muscle stress device will be used. The muscle stress apparatus will provide for preplanned isometric muscle forces from the calf muscles and the arm muscles. The standardized test protocol will include measures of muscle strength, muscle endurance, and muscle fatigability. Pre- and postflight measures only will be taken.

B. Procedure

The test procedure is identified in Figure 1. Two muscles each from the leg and arm will be instrumented and the muscle action potentials recorded. These muscles are the brachial bicep, brachioradialis, gastrocnemius and soleus. The procedure will require about 11 minutes to complete.

C. Data

The EMG data will be recorded on magnetic tape for time and frequency domain analysis at the JSC Cardiovascular Laboratory.

PROCEDURE

A. Apply surface electrodes

B. Seat subject in muscle stress device prepared for muscle stress test

C. Three short (1 to 2 seconds) efforts to determine the maximum voluntary contraction (MVC)

D. Remainder of calf procedure

   1. 10 seconds at 10% MVC - 20 seconds rest

   2. 10 seconds at 20% MVC - 20 seconds rest

   3. 10 seconds at 30% MVC - 20 seconds rest

   4. 60 seconds at 50% MVC

E. Reset muscle stress device for arm stress test

F. Three short (1 to 2 seconds) efforts to determine the maximum voluntary contraction

G. Remainder of arm procedure

   1. 10 seconds at 10% MVC - 20 seconds rest

   2. 10 seconds at 20% MVC - 20 seconds rest

   3. 10 seconds at 30% MVC - 20 seconds rest

   4. 60 seconds at 50% MVC

H. Remove surface electrodes

APPENDIX B

INPUT/OUTPUT MAGNETIC TAPE AND DATA CARD FORMATS

EMG DATA INPUT TAPE RECORD

1 Record #	2 Hours	3 Minutes	4 Seconds	5 Mili-Seconds	6 Subject No.	7 Exp. Month	8 Exp. Day	9 Exp. Year	10 Digital Month	11 Digital Day	12 Digital Year

13 Flight Ref. Day	14 Run No.	15 Analog Tape No.	16 EMG Samp Rate (S/S)	17 Force Samp Rate (S/S)	18 RMS Amplitude EMG Cal	19 Not Used	20 Amount of Force Cal (Lbs)	21 Not Used	22 Not Used	23 Not Used	24 Not Used

25 Not Used	26 EMG Data Samp.	27 Force Data Sample	28 EMG Data Sample	29 EMG Data Sample		→		78 Force Data Sample	79 EMG Data Sample	→	

	129 Force Data Sample	130 EMG Data Sample	→		180 Force Data Sample	181 EMG Data Sample	→		231 Force Data Sample	232 EMG Data Sample	→

	282 Force Data Sample	283 EMG Data Sample	→		333 Force Data Sample	334 EMG Data Sample	→		334 Force Data Sample	335 EMG Data Sample	→

	435 Force Data Sample	436 EMG Data Sample	→		486 Force Data Sample	487 EMG Data Sample	→			534 EMG Data Sample	535 EMG Data Sample

HEADER RECORD FOR PSD OUTPUT

Word #	
1	Subject Number
2	Experiment Month
3	Experiment Day
4	Experiment Year
5	Digitizing Month
6	Digitizing Day
7	Digitizing Year
8	Flight Reference Day
9	Run Number
10	Analog Tape Number
11	Sample Rate for EMG Signal (Samp/sec)
12	Sample Rate for Force Signal (Samp/sec)
13	RMS Amplitude of Sine Wave Cal (in microvolts)
14	Not Used
15	Amount of Force Cal
16	Start Hour
17	Start Minute
18	Start Second
19	Length of Data Slice (sec)
20	Muscle ID Number
21	# of Following Records with PSD Values
22 - 25	Not Used

DATA CARD FORMAT

First Card In Data Card Deck

cc: 1-5	Number of data slices (max = 200)

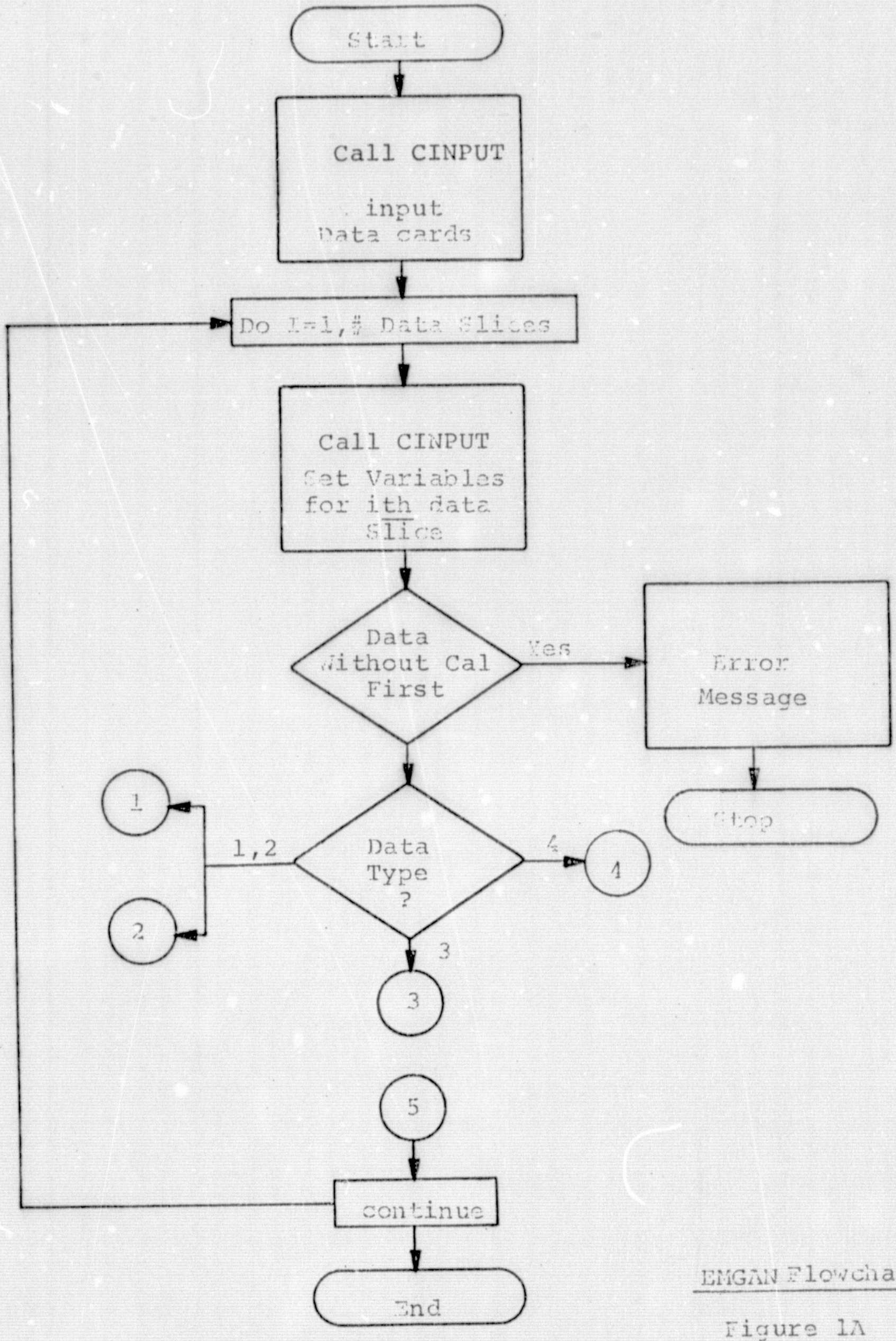
Data Slice Cards

cc: 1-3	File data slice located in (00 to 99)
cc: 4-6	Data Type 1 = Force Cal 2 = EMG Cal 3 = EMG Data 4 = Force Data
cc: 7-9	HR start time of data slice
cc: 10-12	Min start time of data slice
cc: 13-15	Sec start time of data slice
cc: 16-18	Filter switch -1 = No filter 0 = Filter 60 Hz 1 = Filter 60, 180 and 300 Hz
cc: 19-21	Length of data slice in seconds Force data or cal - any integer number up to 120 seconds EMG Cal - 1, 2, 3 or 4 seconds EMG Data - 1, 2, or 4 seconds
cc: 22-24	Plot switch 0 = Plot data 1 = Suppress plotting
cc: 25-27	Print switch 0 = Print data 1 = Suppress printing

cc: 28-30	Averaging interval (force and EMG data cards only)
	Force Data = over any number of seconds up to the length of data slice
	EMG Data = (Frequency smoothing only) any integer number up to 400
cc: 31-33	Force signal gain with respect to force calibration (force cards only
cc: 34-36	Muscle Type (cal data cards only)
	1 = Brachial Bicep
	2 = Brachioradialis
	3 = Gastrocnemius
	4 = Soleus

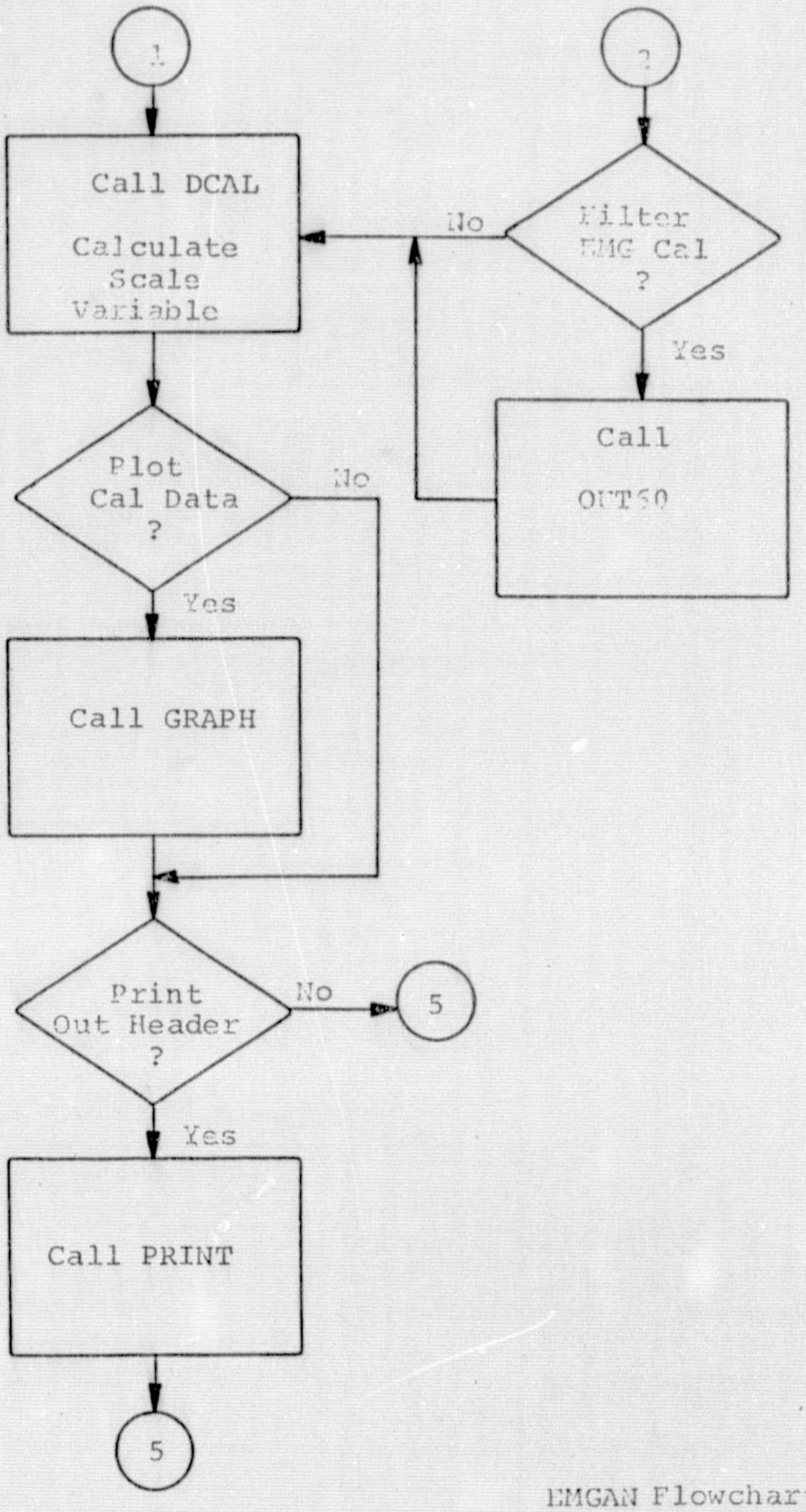
APPENDIX C

MAIN PROGRAM FLOW CHART



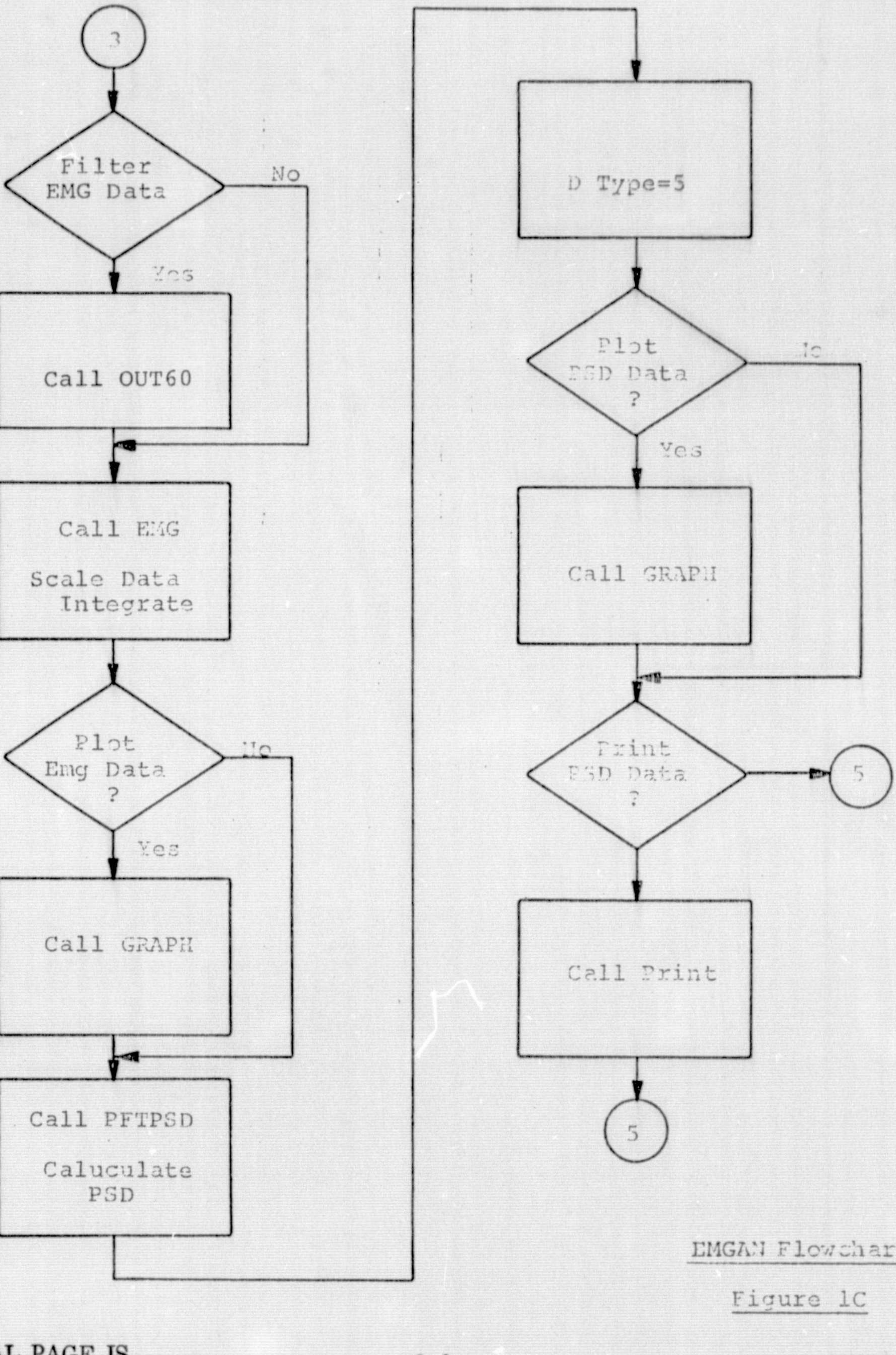
EMGAN Flowchart

Figure 1A



EMGAN Flowchart

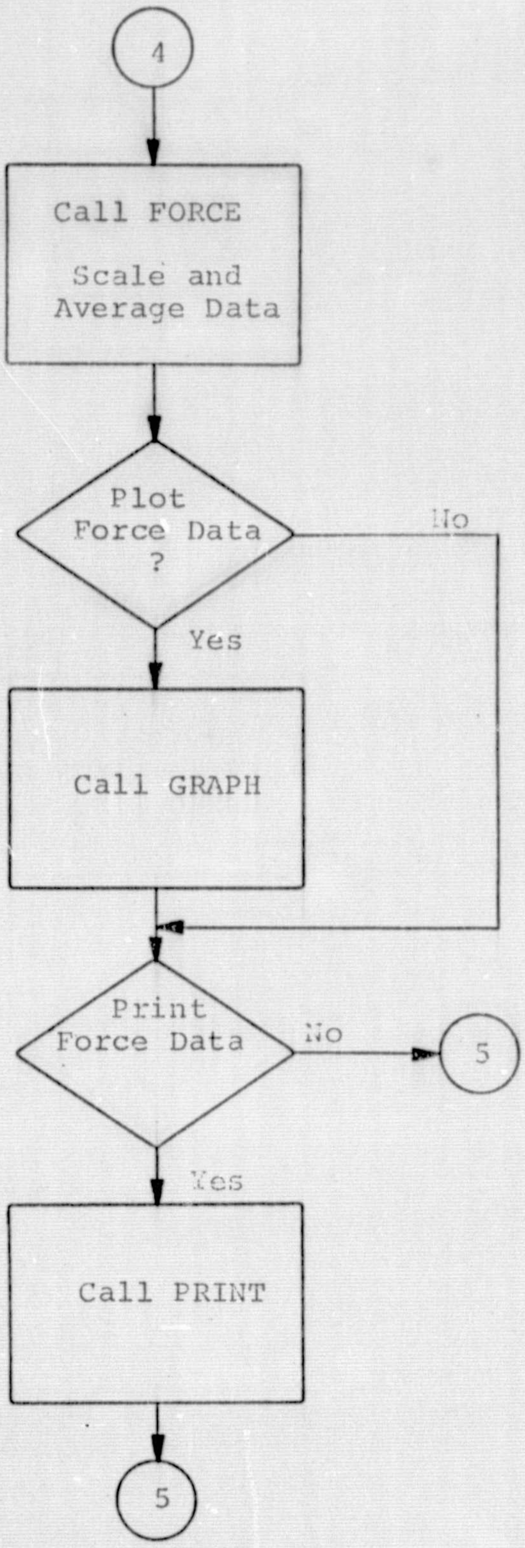
Figure 1B



EMGAN Flowchart

Figure 1C

ORIGINAL PAGE IS
OF POOR QUALITY



EMGAN Flowchart

Figure 1D

APPENDIX D

TABLE OF FORMULAS

TABLE OF FORMULAS

1. Force Calibration Variable (counts/pound)

$$FORSLP = (H/20 - L/20)/FLBCAL$$

where H = sum of samples for one second of force calibration level

L = sum of samples for one second of zero force level

$FLBCAL$ = amount of force calibration in pounds

2. EMG Calibration Variable (counts/microvolt)

$$EMGAL = \sqrt{\sum X_i^2/(NPTS-1)/MAGTUD}$$

where $\sum X_i$ = Sum of data values squared

$NPTS$ = number of EMG cal signal samples

$MAGTUD$ = RMS voltage of EMG cal signal

3. Mean Square (Variance)

$$EMGVAR = \sum X_i^2 /(NPTS-1)$$

where $\sum X_i^2$ = sum of data values squared

$NPTS$ = number of data points

4. Simpson's Rule (without error term)

$$VOLTSEC = H/3\ (2SUMO + 4\ SUME - Data\ (1) + Data\ (n))$$

where H = spacing between the data values

SUMO = sum of the odd numbered data values excluding first data value

SUME = sum of the even numbered data values excluding last data value

Data (1) = first data value

Data (n) = last data value

5. Scaling of EMG Data (microvolts)

SEDATA(i) = (USEDATA(i) - OFFSET)/EMGCAL

where	USEDATA(i)	= i^{th} unscaled data value
	OFFSET	= DC value of EMG data array
	EMGVAL	= EMG calibration value

6. Scaling of Force Data (pounds)

SFDATA(i) = (USFDATA(i) - BASE) * GAIN/FORSLP

where	USFDATA(i)	= i^{th} unscaled force data value
	BASE	= zero force level
	GAIN	= ratio of force calibration gain to force data gain
	FORSLP	= force calibration variable

7. Discrete Fourier Transform

$$X(n) = \sum_{K=0}^{N-1} x(k)\,(\mathrm{EXP}\,(-j2\,\pi/N))^{nk} \qquad n = 0,\ 1, \ldots N-1$$

where x (k) = k^{th} value of untransformed data

n = number of data points

8. Power Spectral Density (magnitude)

$$PSD(n) = 8H/NPTS\ [Re(X(n))^2 + Im(X(n))^2]$$

where H = spacing between data samples

NPTS = number of data samples

Re(X(n)) = real part of nth Fourier transform value

Im(X(n)) = imaginary part of nth Fourier transform value

9. Expected (Mean) Value of PSD

$$EXPVAL = \sum F\ (PSD(n) - PERCNT(n)/100)$$

where PSD (n) = value of nth power spectral density bandwidth

PERCNT (n) = percent of total power contributed by nth bandwidth

F = center frequency of nth bandwidth

10. Standard Deviation of PSD

$$STDEV = \sqrt{\sum (F-EXPVAL)^2 * (PERCNT(n)/100)}$$

where EXPVAL = expected value of PSD

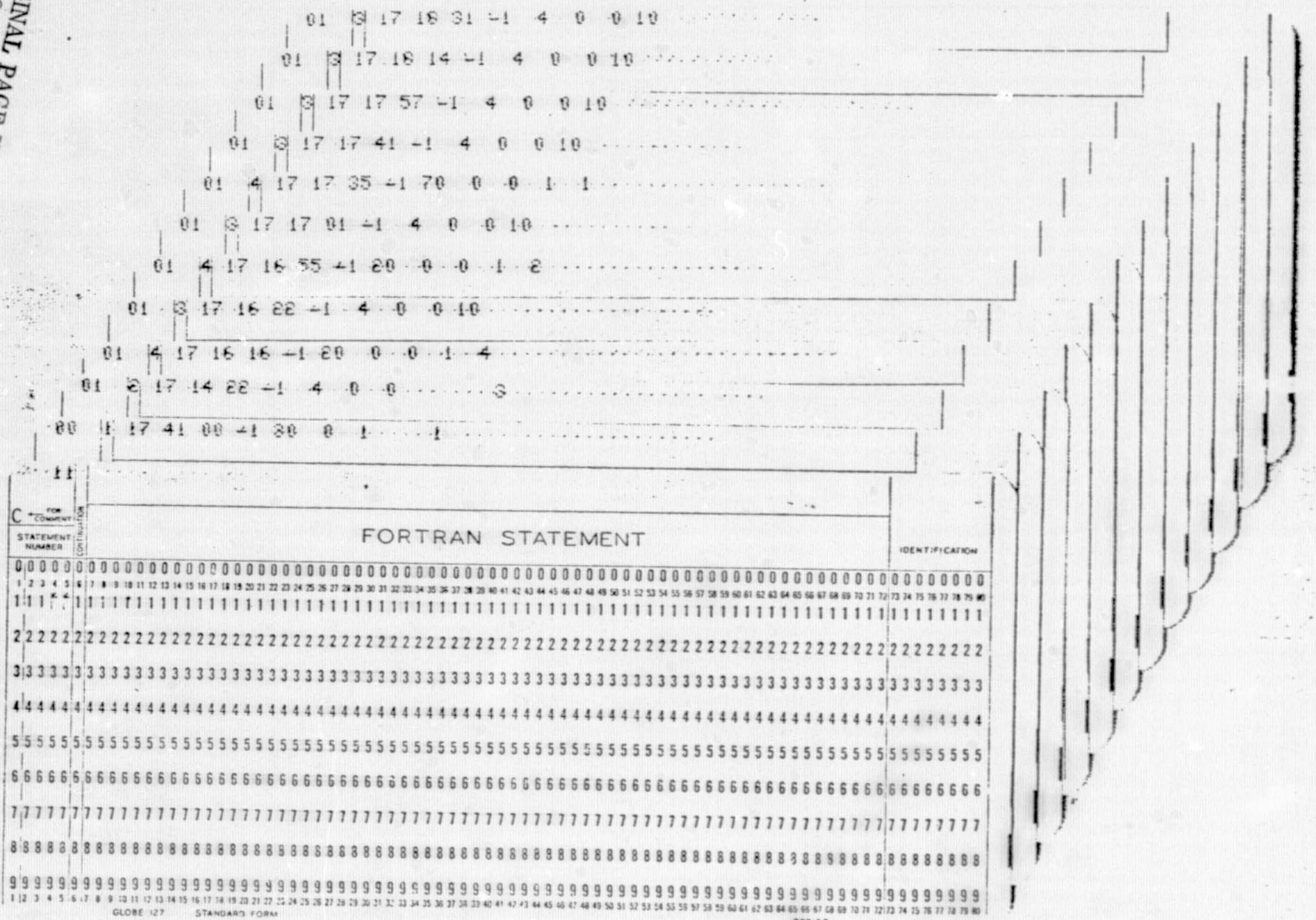
PERCNT (n) = percent of total power contributed by nth bandwidth

F = center frequency of nth bandwidth

APPENDIX E

EXAMPLE OF EMGAN INPUT CARD DECK WITH PLOTTED AND PRINTED OUTPUT

ORIGINAL PAGE IS
OF POOR QUALITY



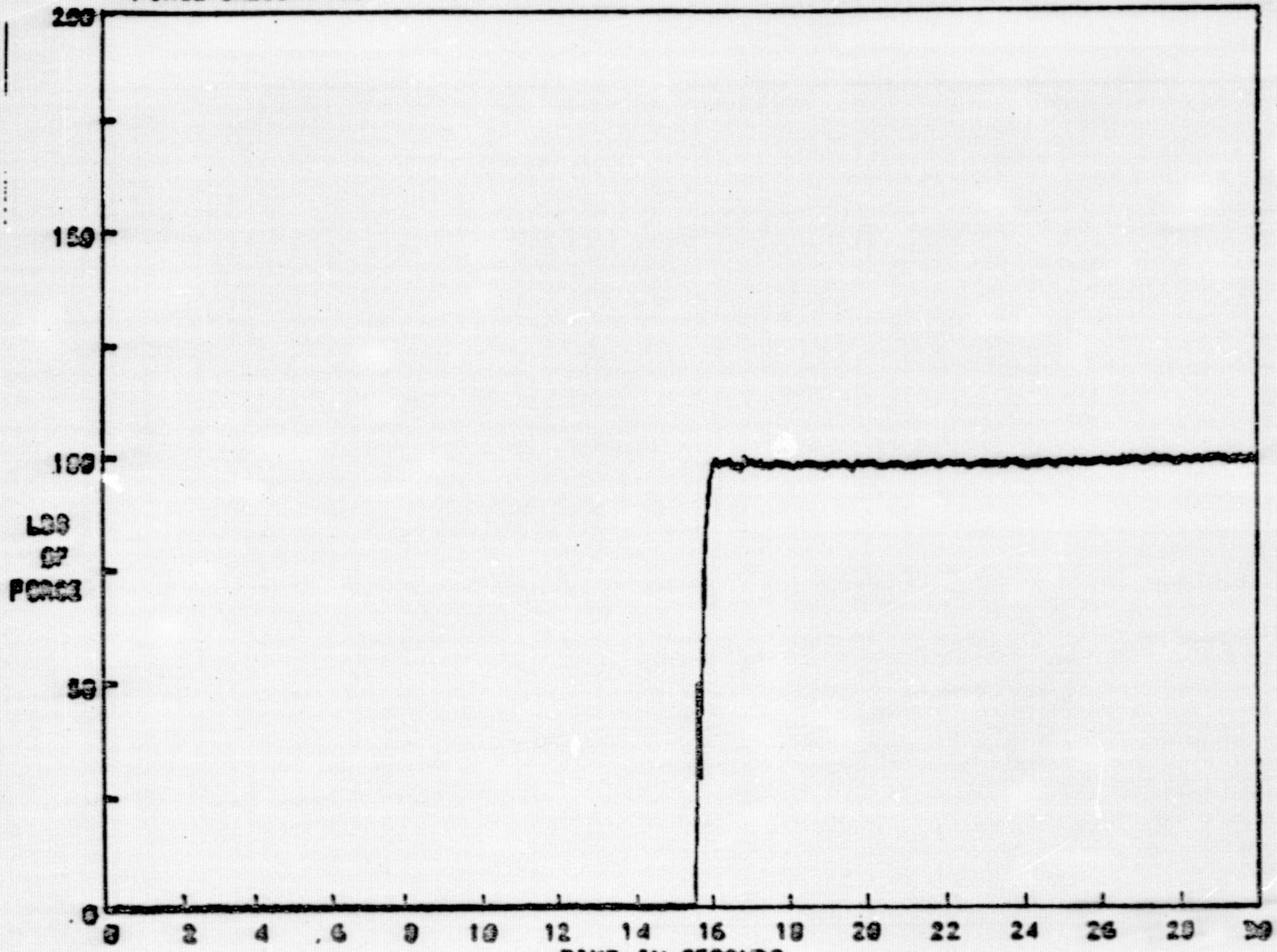
```
01  3 17 18 31 -1  4  0  0 10
01  3 17 18 14 -1  4  0  0 10
01  3 17 17 57 -1  4  0  0 10
01  3 17 17 41 -1  4  0  0 10
01  4 17 17 35 -1 70  0  0  1  1
01  3 17 17 01 -1  4  0  0 10
01  4 17 16 55 -1 20  0  0  1  2
01  3 17 16 22 -1  4  0  0 10
01  4 17 16 16 -1 20  0  0  1  4
01  2 17 14 22 -1  4  0  0     3
00  1 17 41 00 -1 30  0  1     1
11
```

```
 **** DATA CARDS ****


NO. OF DATA SLICES=    11


    DATA SLICES


00   1 17 41  0 -1 30   0   1   0   1   0   0   0   0
01   2 17 14 22 -1  4   0   0   0   0   3   0   0   0
01   4 17 16 16 -1 20   0   0   1   4   0   0   0   0
01   3 17 16 22 -1  4   0   0  10   0   0   0   0   0
01   4 17 16 55 -1 20   0   0   1   2   0   0   0   0
01   3 17 17  1 -1  4   0   0  10   0   0   0   0   0
01   4 17 17 35 -1 70   0   0   1   1   0   0   0   0
01   3 17 17 41 -1  4   0   0  10   0   0   0   0   0
01   3 17 17 57 -1  4   0   0  10   0   0   0   0   0
01   3 17 18 14 -1  4   0   0  10   0   0   0   0   0
01   3 17 18 31 -1  4   0   0  10   0   0   0   0   0
```

```
                         CARDIOVASCULAR LABORATORY

                         EMG DATA PROCESSING PROGRAM
                         ---------------------------


                         *** HEADER INFORMATION***

SUBJECT NO.:          44          EXPERIMENT DATE:       6/28/75          FLIGHT REFERENCE DAY:  F-15

RUN NO.:              3           ANALOG TAPE NO.:         9              DIGITIZING DATE:        6/29/75

MUSCLE:            GASTROCNEMIUS

EMG SAMPLE RATE (SAMP/SEC): 1000  FORCE SAMPLE RATE (SAMP/SEC):   20


                          ** CALIBRATION DATA **

FORCE CAL
---------

CAL WEIGHT (LBS):   98.0          COUNTS/LB:     39.    AVG. BASELINE COUNT:  -3662.    CAL GAIN:   1.

EMG CAL
-------

RMS AMPLITUDE (MICROVOLTS):  350.   COUNTS/MICROVOLT: 6.44
```

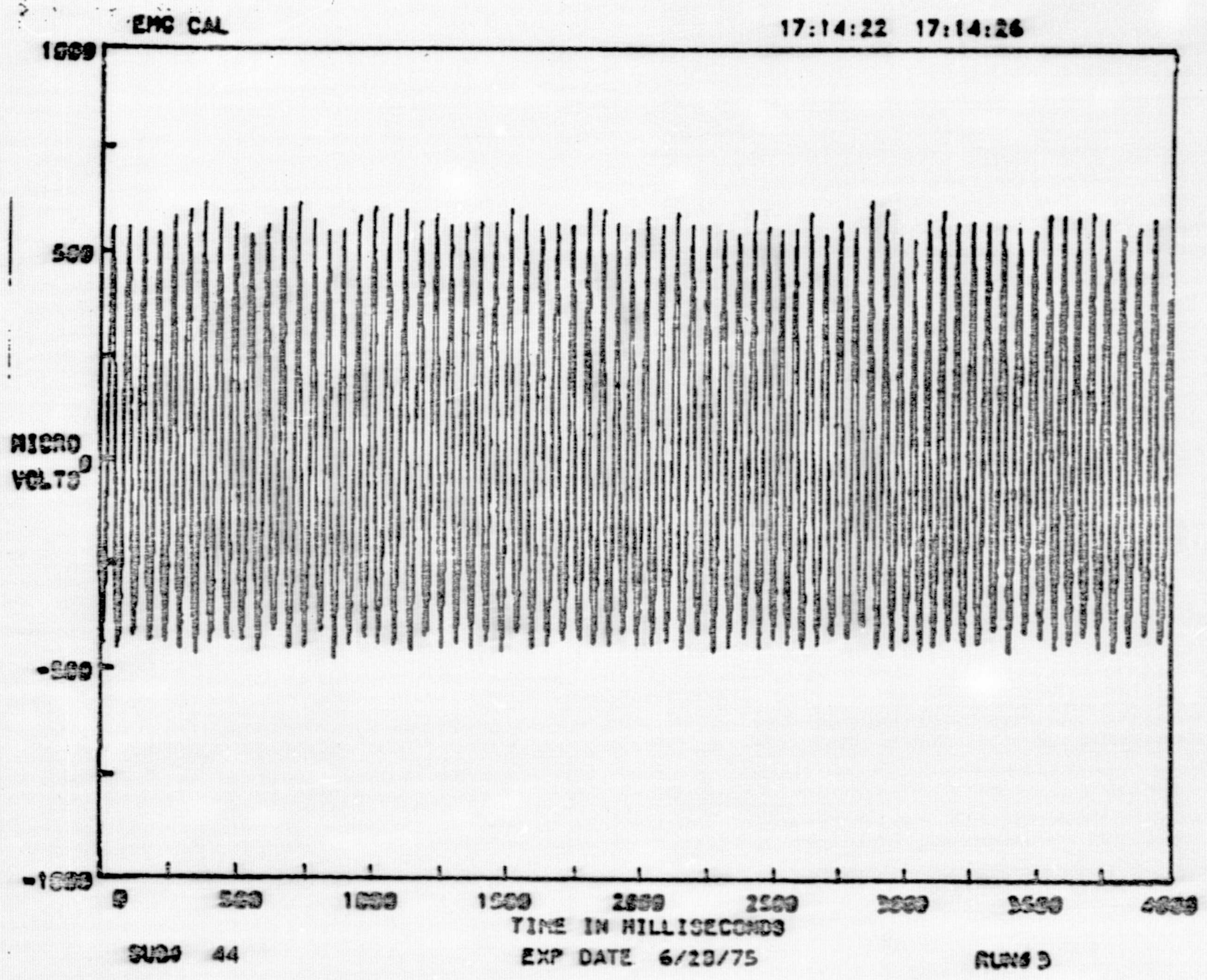
```
                    *** FORCE DATA ***

TIME- 17:16:16          DATA LENGTH (SECS)-  20          AVERAGE FORCE INTERVAL (SECS)- 1.

                 INTERVAL     AVERAGE FORCE (LBS)
                 --------     -------------------

                     1              0.0
                     2              0.0
                     3              0.0
                     4              0.1
                     5             27.7
                     6             41.1
                     7             41.9
                     8             41.5
                     9             41.4
                    10             40.9
                    11             42.5
                    12             42.2
                    13             41.3
                    14             41.6
                    15             42.3
                    16             41.6
                    17             21.7
                    18              0.0
                    19             -0.0
                    20              0.0
                    *** POWER SPECTRAL DENSITY OF EMG DATA ***

START TIME- 17:16:22       DATA LENGTH (SECS)-   4           INTEGRATED EMG (MICROVOLT*SEC)-   0.2676E  3

BANDWIDTH (HZ)- 9.766      NORMALIZED STANDARD ERROR- 0.158  INTEGRATED PSD (MICROVOLT**2)-    0.7890E  4

MEAN (HZ)- 149.2           STANDARD DEVIATION(HZ)-  80.8     EMG VARIANCE (MICROVOLT**2)-      0.8014E  4

                   FREQ          PSD            PSD         % OF          CUM %
                   (HZ)      (MMV**2/HZ)        NORM        TOTAL         TOTAL
                   ----      -----------        ----        -----         -----

                   4.88         1.398          0.0306        0.17          0.17
                  14.65         6.766          0.1482        0.84          1.01
                  24.41        15.152          0.3319        1.88          2.89
                  34.18        23.153          0.5072        2.87          5.75
                  43.95        18.558          0.4065        2.30          8.05
                  53.71        30.225          0.6621        3.74         11.79
                  63.48        34.020          0.7452        4.21         16.00
                  73.24        27.169          0.5951        3.36         19.36
                  83.01        45.652          1.0000        5.65         25.01
                  92.77        32.585          0.7138        4.03         29.05
                 102.54        39.469          0.8646        4.89         33.93
                 112.30        42.109          0.9224        5.21         39.14
                 122.07        40.914          0.8962        5.06         44.21
                 131.84        45.400          0.9945        5.62         49.83
                 141.60        39.421          0.8635        4.88         54.71
                 151.37        43.597          0.9550        5.40         60.10
                 161.13        36.991          0.8103        4.58         64.68
                 170.90        34.035          0.7455        4.21         68.89
                 180.66        20.588          0.4510        2.55         71.44
                 190.43        21.723          0.4758        2.69         74.13
                 200.20        23.014          0.5041        2.85         76.98
                 209.96        27.384          0.5998        3.39         80.37
                 219.73        20.216          0.4428        2.50         82.87
                 229.49        15.688          0.3436        1.94         84.81
                 239.26        17.412          0.3814        2.16         86.97
                 249.02        13.422          0.2940        1.66         88.63
                 258.79         9.829          0.2153        1.22         89.84
                 268.55        10.397          0.2277        1.29         91.13
                 278.32         9.821          0.2151        1.22         92.35
                 288.09         6.011          0.1317        0.74         93.09
                 297.85         8.662          0.1897        1.07         94.16
                 307.62         7.959          0.1743        0.99         95.15
                 317.38         6.416          0.1405        0.79         95.94
                 327.15         5.909          0.1294        0.73         96.67
                 336.91         7.114          0.1558        0.88         97.55
                 346.68         5.590          0.1224        0.69         98.25
                 356.45         3.670          0.0803        0.45         98.70
                 366.21         4.024          0.0881        0.50         99.20
                 375.98         2.574          0.0564        0.32         99.52
                 385.74         2.234          0.0489        0.28         99.79
                 395.51         1.670          0.0366        0.21        100.00
```

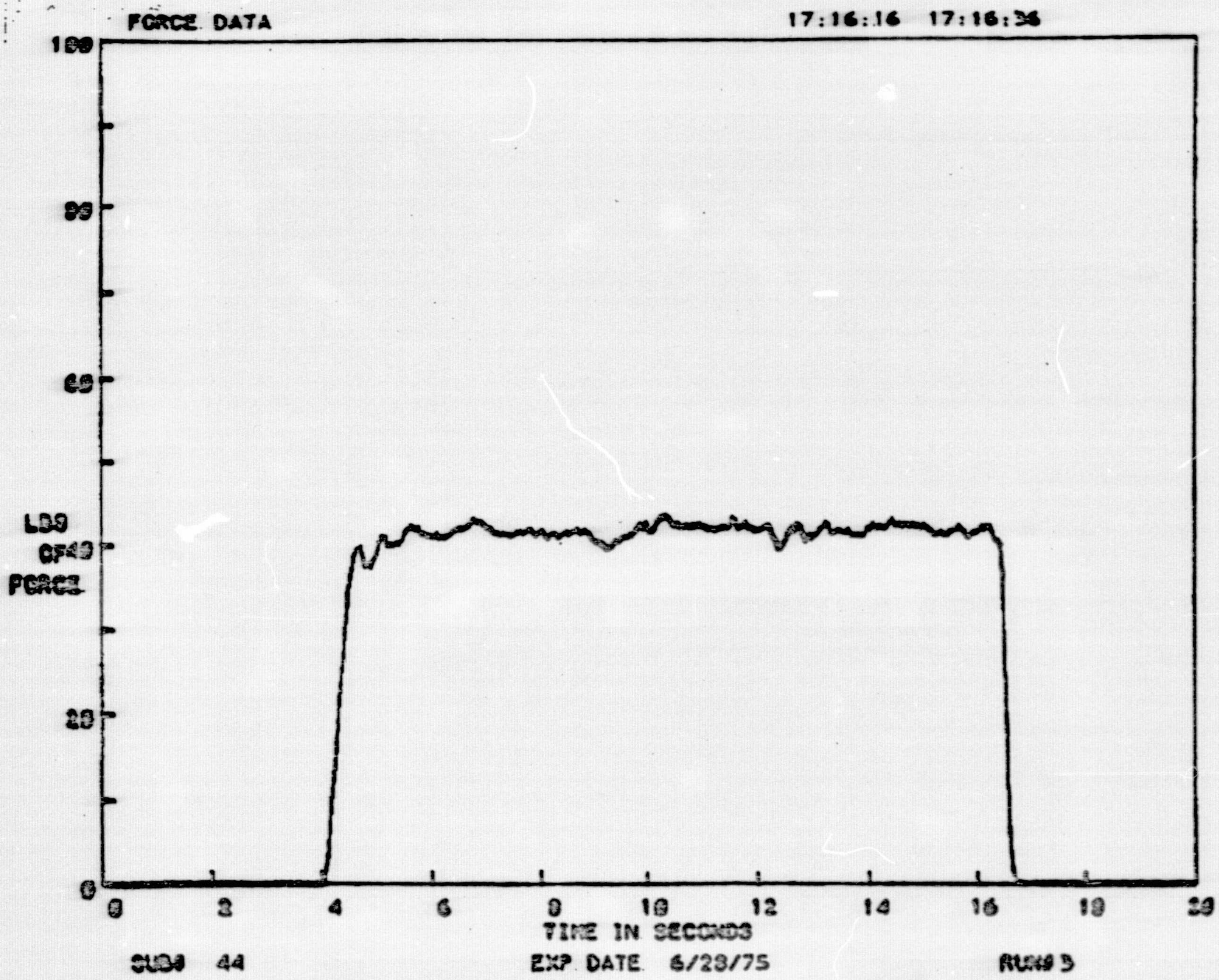
FORCE CALIBRATIO 17:41: 0 17:41:30

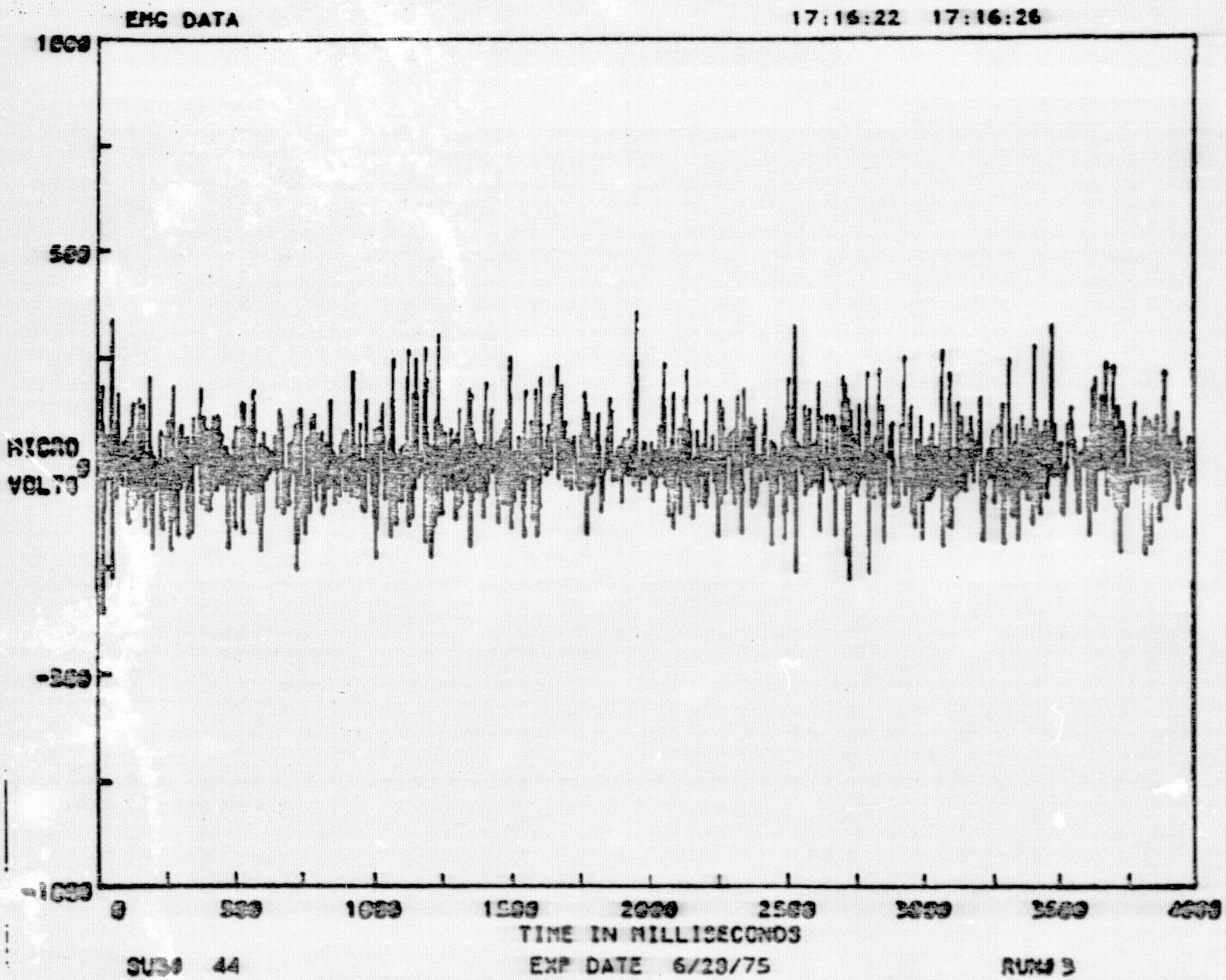


SUB# 44 EXP DATE 6/23/75 RUN# 3

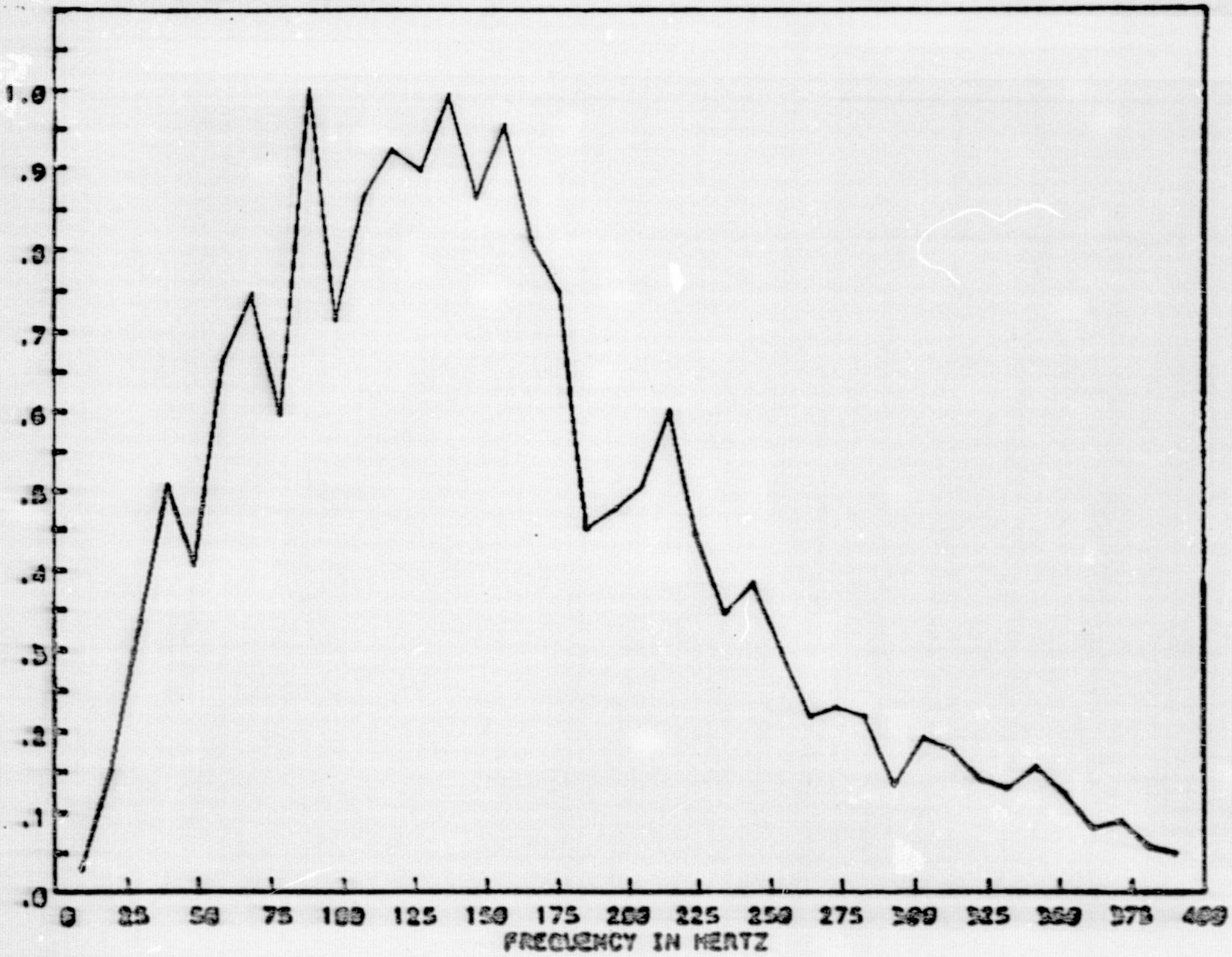
AMOUNT OF FORCE CAL (LBS) 98


EMG CAL
17:14:22 17:14:26
1000
500
MICRO VOLTS 0
-500
-1000
0
500
1000
1500
2000
2500
3000
3500
4000
TIME IN MILLISECONDS
SUB# 44
EXP DATE 6/23/75
RUN# 3


FORCE DATA
17:16:16 17:16:36
100
80
60
LBS
OF 40
FORCE
20
0
0
2
4
6
8
10
12
14
16
18
20
TIME IN SECONDS
SUBJ 44
EXP DATE 6/23/75
RUN# 3


EMG DATA
17:16:22 17:16:26
1000
500
0
-500
-1000
MICRO VOLTS
0
500
1000
1500
2000
2500
3000
3500
4000
TIME IN MILLISECONDS
SUBJ 44
EXP DATE 6/23/75
RUN# 3

POWER SPECTRAL DENSITY 17:16:22 17:16:26



SUBJ 44 EXP DATE 6/28/75 RUN# 3

BANDWIDTH (HZ) = 9.75 NORMALIZED STANDARD ERROR= .150

APPENDIX F

SIGMA SOURCE LISTING AND LOAD MAP

```
]JOB
]ASS UO=6
]EDITOR
-LIST
-GEN
C***********************************************************************
C
C
C      PROGRAM:     EMGAN
C
C      AUTHOR:      WILLIAM N. HURSTA
C
C      PURPOSE:     TO PROVIDE  FIRST ORDER REDUCTION OF EMG AND FORCE DATA.
C
C***********************************************************************
C
C
      DIMENSION IVARSW(10)
      INTEGER FILNOW,DTYPE,FILNUM,FILTSW,PLOTSW
      INTEGER FCALSW ,ECALSW,WHOA,PRNTSW
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      EQUIVALENCE (DTYPE,IVARSW(1)),(FILNUM,IVARSW(2))
      EQUIVALENCE (FILTSW,IVARSW(3)),(NSECS,IVARSW(4))
      EQUIVALENCE (PLOTSW,IVARSW(5)),(PRNTSW,IVARSW(6))
      EQUIVALENCE (ISPAN,IVARSW(7)),(IFGAIN,IVARSW(8))
      FILNOW=1
      FCALSW=0
      ECALSW=0
      WHOA=0
      NSLICE=0
C
C      GO GET NUMBER OF DATA SLICES FOR PRESENT RUN
C
      CALL SEGLD (1)
      CALL CINPUT (IVARSW,NSLICE,I,START)
C
C     MAIN PROGRAM DO LOOP
```

```
C
      DO 200 I=1,NSLICE
C
C     GO GET DATA CARD INFO FOR PRESENT DATA SLICE
C
      CALL SEGLD (1)
      CALL CINPUT (IVARSW,NSLICE,I,START)
      SPAN=FLOAT(ISPAN)
      IF(DTYPE.EQ.4) FDGAIN=FLOAT(IFGAIN)
C
C      CHECK TO SEE IF CALIBRATION DATA ACQUIRED BEFORE PROCESSING FIRST DATA
C      SLICE
C
      IF(DTYPE.EQ.1) FCALSW=1
      IF(DTYPE.EQ.2) ECALSW=1
      IF(DTYPE.EQ.3.AND.ECALSW .NE.1) WHOA=1
      IF(DTYPE.EQ.4.AND.FCALSW.NE.1) WHOA=1
      IF(WHOA.EQ.1) GO TO 90
C
C      GO GET THE DATA FROM TAPE
C
      CALL SEGLD(2)
      CALL TINPUT(DTYPE,FILNOW,FILNUM,START,NSECS)
C
C      DECIDE WHERE TO GO FOR EACH DATA TYPE.
C
      GO TO (20,10,30,70),DTYPE
C
C      FILTER 60 HZ  FOR EMG CAL?
C
   10 IF(FILTSW.EQ.-1) GO TO 20
      CALL OUT60 (DTYPE,NSECS,FILTSW)
C
C      CALCULATE CALIBRATION VARIABLES OF EMG OR FORCE DATA.
C
   20 CALL SEGLD(3)
      CALL DCAL (DTYPE,NSECS,IFGAIN)
C
```

```
C       PLOT EMG OR FORCE CAL DATA?
C
   25 IF(PLOTSW.EQ.1) GO TO 26
      CALL SEGLD(4)
      CALL GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
C
C       PRINT OUT HEADER?
C
   26 IF(PRNTSW.EQ.1) GO TO 200
      GO TO 90
C
C       FILTER 60 HZ  FOR EMG DATA SLICE?
C
   30 IF(FILTSW.EQ.-1) GO TO 40
      CALL OUT60 (DTYPE,NSECS,FILTSW)
C
C       SCALE EMG DATA, CALCULATE INTEGRATED VALUE AND MEAN SQUARE VALUE
C
   40 CALL SEGLD (5)
      CALL EMG (NSECS,VOLTSEC)
C
C       PLOT EMG DATA ON COMPUTEK?
C
      IF(PLOTSW.EQ.1) GO TO 50
      CALL SEGLD(4)
      CALL GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
C
C       TAKE THE FOURIER TRANSFORM OF THE DATA AND FIND PSD UP TO 400 HZ.
C
   50 CALL SEGLD (6)
      CALL FFTPSD (NSECS,SPAN,PSDSUM,STDERR)
      DTYPE=5
C
C       PLOT THE PSD ON THE COMPUTEK?
C
      IF(PLOTSW.EQ.1) GO TO 60
      CALL SEGLD(4)
      CALL GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
```

```
C
C       PRINT OUT RESULTS ON LINE PRINTER?
C
   60 IF(PRNTSW.EQ.1) GO TO 200
      GO TO 90
C
C       SCALE FORCE DATA FOR PLOTTING AND AVERAGE IT FOR PRINTING.
C
   70 CALL SEGLD (5)
      CALL FORCE (DTYPE,NSECS,SPAN,FDGAIN)
C
C       PLOT FORCE DATA ON COMPUTEK?
C
      IF(PLOTSW.EQ.1) GO TO 80
      CALL SEGLD(4)
      CALL GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
C
C       PRINT OUT FORCE DATA ON LINE PRINTER?
C
   80 IF(PRNTSW.EQ.1) GO TO 200
   90 CALL SEGLD(7)
      CALL PRINT (DTYPE,START,NSECS,SPAN,VOLTSEC,PSDSUM,STDERR,WHOA)
  200 CONTINUE
      END
      SUBROUTINE CINPUT (IVARSW,NSLICE,I,START)
C
C       SUBROUTINE CINPUT AQUIRES THE CARD DATA AND STORES IT ON A RAD FILE
C
C         ARGUMENTS:  IVARSW- ARRAY HOLDING THE VARIABLES AND SWITCHES FOR EACH
C                             DATA SLICE (SEE EQUIVALENCE STATEMENTS IN MAIN
C                             PROGRAM)
C
C                     NSLICE- NUMBER OF DATA SLICES IN PRESENT RUN
C
C                     I     - IDENTIFIES PRESENT DATA SLICE BEING PROCESSED
C
C                     START - BEGIN TIME OF PRESENT DATA SLICE
C
```

```
      DIMENSION IVARSW(10),ICARD(200,15)
      DIMENSION IRDIO(10),IRDF(6)
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      EQUIVALENCE (ICARD,DATA(1))
      DATA IRDIO(1) /'RD'/
      DATA IRDF /Z4000,'RD',1,6000,0,0/
      CALL DEFINE (IRDF,IRDF)
      IF(NSLICE.EQ.0) GO TO 30
      GO TO 60
C
C       INPUT NUMBER OF DATA SLICES AND READ DATA SLICE INFO FOR PRESENT RUN
C
   30 READ(105,20) NSLICE
      WRITE(108,35) NSLICE
   20 FORMAT(I5)
   35 FORMAT(1H1,' **** DATA CARDS ****',///,' NO. OF DATA SLICES=',I5,/
     @//,5X,'DATA SLICES',//)
      IF(NSLICE.GT.200) GO TO 100
      DO 50 J=1,NSLICE
      READ(105,40) (ICARD(J,K),K=1,15)
   40 FORMAT(15I3)
      WRITE(108,40) (ICARD(J,K),K=1,15)
   50 CONTINUE
      CALL QOPEN(IRDIO,ICARD,3000,1,0)
      CALL QWRITE(IRDIO,3000,INDIC)
      CALL QCLOSE (IRDIO,2)
      GO TO 200
C
C       INITIALIZE VARIABLES AND SWITCHES FOR EACH DATA SLICE
C
   60 CALL QOPEN(IRDIO,ICARD,3000,0,0)
      CALL QREAD(IRDIO,3000,INDIC)
      CALL QCLOSE (IRDIO,2)
      IVARSW(1)=ICARD(I,2)
      IVARSW(2)=ICARD(I,1)+1
      START=3600.*FLOAT(ICARD(I,3))+60.*FLOAT(ICARD(I,4))+FLOAT(ICARD(I,
     @5))
      IVARSW(3)=ICARD(I,6)
      IVARSW(4)=ICARD(I,7)
      IVARSW(5)=ICARD(I,8)
      IVARSW(6)=ICARD(I,9)
      IVARSW(7)=ICARD(I,10)
      IVARSW(8)=ICARD(I,11)
      IHEAD(16)=ICARD(I,3)
      IHEAD(17)=ICARD(I,4)
      IHEAD(18)=ICARD(I,5)
      IHEAD(19)=ICARD(I,7)
      IF(IVARSW(1)-2) 90,90,200
   90 IHEAD(20)=ICARD(I,12)
      GO TO 200
  100 WRITE(108,110) NSLICE
  110 FORMAT(///,' ERROR...READ NUMBER OF DATA SLICES TO BE',I6,'.  CANN
     @OT HAVE MORE THAN 200.')
      STOP
  200 RETURN
      END
      SUBROUTINE TINPUT(DTYPE,FILNOW,FILNUM,START,NSECS)
C
C      SUBROUTINE TINPUT READS FORCE AND EMG DATA IN FROM TAPE.
C
C        CALLING ARGUMENTS:  DTYPE - DATA TYPE
C
C                            FILNOW- FILE ON TAPE CURRENTLY BEING ACCESSED
C
C                            FILNUM- FILE ON TAPE WHERE DESIRED DATA IS LOCATED
C
C                            START - TIME IN TOTALED SECONDS OF BEGINNING OF
C                                    DESIRED DATA SLICE
C
C                            NSECS - NUMBER OF SECONDS OF DATA TO BE READ .
C
C
      INTEGER FILMOVE,FILNUM,DTYPE,FILNOW
      DIMENSION INTP(10),IARRAY(535)
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      EQUIVALENCE (IARRAY,DATA(8230))
      INTP(1)=6
      CALL QOPEN (INTP,IARRAY,535,2,0)
C
C      POSITION TAPE AT THE BEGINNING OF THE DESIRED DATA FILE
C
      IF(FILMOVE.LT.0) FILMOVE=FILMOVE-1
      CALL QFSKIP (INTP,FILMOVE,INDIC)
      GO TO (300,5),INDIC/4
      IF(FILMOVE.LT.0) CALL QFSKIP (INTP,1,INDIC)
    5 IF(FILMOVE.LT.0) FILMOVE=FILMOVE+1
      FILMOVE=FILNUM-FILNOW
      FILNOW=FILNOW+FILMOVE
C
C      POSITION TAPE AT THE BEGINNING OF THE DATA SLICE.
C
      CALL FIND (FILNUM,START,INTP)
C
C      DECIDE WHERE TO GO FOR EACH DATA TYPE.
C
      GO TO (10,50,50,10),DTYPE
C
C      ACQUIRE FORCE CAL OR FORCE DATA
C
   10 JJ=1
      DO 30 J=1,2*NSECS
      CALL QREAD (INTP,535,INDIC)
      IF(INDIC.NE.0) GO TO 200
      DO 20 K=27,535,51
      DATA(JJ)=FLOAT(IARRAY(K)/8)
      JJ=JJ+1
   20 CONTINUE
   30 CONTINUE
      IF(DTYPE.EQ.1) GO TO 1000
      GO TO 150
C
C      ACQUIRE EMG CAL OR EMG DATA
C
   50 JJ=1
```

```
      DO 80 J=1,2*NSECS+1
      CALL QREAD (INTP,535,INDIC)
      IF(INDIC.NE.0) GO TO 200
      DO 70 K=1,10
      L=25+K+50*(K-1)
      DO 60 KK=L,L+50
      IF(KK.EQ.L+1) GO TO 60
      DATA(JJ)=FLOAT(IARRAY(KK)/8)
      JJ=JJ+1
   60 CONTINUE
   70 CONTINUE
   80 CONTINUE
      IF(DTYPE.EQ.2) GO TO 1000
  150 IBACK=-(2*NSECS+2)
      CALL QSKIP(INTP,IBACK,INDIC)
      GO TO 1000
C
C      ERROR MESSAGES
C
  200 FILNUM=FILNUM-1
      WRITE(108,210) INDIC,DTYPE,FILNUM
  210 FORMAT(///,' ERROR...INDIC=',I4,'DURING INPUT OF DATA TYPE',I4,' L
     @OCATED IN FILE',I4)
      STOP
  300 FILNUM=FILNUM-1
      WRITE(108,310) FILNUM
  310 FORMAT(///,' FATAL ERROR...EOT ENCOUNTERED DURING SEARCH FOR FILE'
     @,I3)
      STOP
 1000 CONTINUE
      CALL QCLOSE (INTP,0)
      RETURN
      SUBROUTINE FIND (FILNUM,START,INTP)
C
C      SUBROUTINE FIND POSITIONS THE TAPE AT THE BEGINNING OF A DATA SLICE
C
C        CALLING ARGUMENTS:  FILNUM- FILE ON TAPE WHERE DATA SLICE IS LOCATED
C
```

```
C                               START - TIME IN TOTALED SECONDS OF THE BEGINNING OF
C                                       DESIRED DATA SLICE.
C
C                               INTP  - CONTROL BLOCK FOR QINOUT TAPE ROUTINES
C
      DIMENSION INTP(10),IARRAY(535)
      INTEGER FILNUM
      REAL NOW
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      EQUIVALENCE(IARRAY,DATA(8230))
      IPASS=0
C
C      READ EACH RECORD AND COMPARE ITS TIME LABEL WITH THE START TIME.
C
   10 CALL QREAD (INTP,535,INDIC)
      IF(INDIC.NE.0) GO TO 20
      NOW=3600.*ABS(FLOAT(IARRAY(2)))+60.*ABS(FLOAT(IARRAY(3)))+ABS(FLOA
     @T(IARRAY(4)))
      IF(NOW.NE.START) GO TO 10
C
C      ACQUIRE THE HEADER INFORMATION FOR THE PRESENT DATA SLICE.
C
      DO 15 J=1,15
      IHEAD(J)=IARRAY(J+5)
   15 CONTINUE
      CALL QSKIP(INTP,-1,INDIC)
      GO TO 100
C
C      DID NOT FIND THE START TIME BEFORE ENCOUNTERING AN EOF. IF ONLY ONE PASS
C      HAS BEEN MADE THRU THE DATA, BACK UP AND TRY AGAIN. OTHERWISE,STOP]
C
   20 IF(IPASS.EQ.1) GO TO 40
      CALL QFSKIP(INTP,-2,INDIC)
      IF(INDIC.EQ.8) GO TO 35
      CALL QFSKIP (INTP,1,INDIC)
   35 IPASS=1
      GO TO 10
   40 FILNUM=FILNUM-1
```

```
      WRITE(108,50) FILNUM
   50 FORMAT(///,'_CANNOT FIND START TIME IN FILE',I4)
      STOP
  100  CONTINUE
      RETURN
      END
      SUBROUTINE OUT60 (DTYPE,NSECS,FILTSW)
C
C      SUBROUTINE OUT60 IS A DIGIT NOTCH FILTER. DEPENDING ON THE DATA CARD
C        REQUEST, ONLY 60 HZ IS REMOVED OR ALL HARMONICS OF 60 HZ UP TO 360 HZ
C
C        CALLING ARGUMENTS:   DTYPE - TYPE OF DATA
C                             NSECS - NUMBER OF SECS OF DATA TO BE FILTERED
C
C                             FILTSW- SWITCH WHICH INDICATES WHETHER ONLY 60 HZ
C                                     IS TO BE REMOVED OR ALL HARMONICS OF 60 HZ
C
C
      INTEGER DTYPE, FILTSW
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      EQUIVALENCE (IRATE,IHEAD(11))
      LOOPS=1
      TWOPI=6.28318530717
      IF(FILTSW.EQ.1) LOOPS=5
      NPTS=1024*NSECS
      DO 30 J=1,LOOPS,2
      A=0.0
      B=0.0
      W=(60*J)*TWOPI/FLOAT(IRATE)
      DO 10 K=1,NPTS
      A=A+DATA(K)*COS(K*W)
      B=B+DATA(K)*SIN(K*W)
   10 CONTINUE
      A=A*2./FLOAT(NPTS)
      B=B*2./FLOAT(NPTS)
      DO 20 K=1,NPTS
      DATA(K)=DATA(K)-A*COS(K*W)-B*SIN(K*W)
   20 CONTINUE
  30 CONTINUE
 100 RETURN
     END
     SUBROUTINE DCAL(DTYPE,NSECS,IFGAIN)
C
C     SUBROUTINE DCAL CALCULATES THE SCALE FACTORS FOR THE FORCE AND EMG DATA
C       THE CALIBRATION CONSTANT FOR INTEGRATED EMG AREA IS ALSO FOUND.  EMG
C     THE CAL DATA IS THEN CONVERTED INTO UNITS OF POUNDS OR MICROVOLTS FOR
C     PLOTTING ON THE COMPUTEK TERMINAL.
C
C       CALLING ARGUMENTS:  DTYPE - TYPE OF DATA
C
C                           NSECS - LENGTH OF CAL DATA
C
C
      DIMENSION HIGH(10),LOW(10),POUNDS(2600)
      INTEGER DTYPE,FLBCAL
      REAL LOWSUM
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      EQUIVALENCE (FLBCAL,IHEAD(15)),(IRATE,IHEAD(11))
      EQUIVALENCE (MAGTUD,IHEAD(13)),(ICPS,IHEAD(14))
      EQUIVALENCE (POUNDS,DATA(1))
      IF(DTYPE.EQ.2) GO TO 100
C
C       FIND FORCE CALIBRATION VARIABLES
C
      CAL(1,1)=FLOAT(FLBCAL)
      CAL(1,4)=FLOAT(IFGAIN)
C
C       LOOK FOR JUMP IN DATA TO INDICATE FORCE CAL.
C
      BEGIN=DATA(80)
      DO 10 J=81,20*NSECS-80
      IF(DATA(J).GT.BEGIN+500.) GO TO 30
   10 CONTINUE
      WRITE (108,20)
   20 FORMAT(///,'COULD NOT FIND THE FORCE CAL.___CHECK DATA SLICE TIMES
     @')
```

```
      STOP
C
C      FOUND THE JUMP, NOW COMPUTE THE AVERAGE BEFORE AND AFTER THE JUMP.
C      CAL WEIGHT IN COUNTS = AVERAGE AFTER - AVERAGE BEFORE
C      FORCE SLOPE = COUNTS/POUND
C      ZERO FORCE  = AVERAGE COUNTS JUST BEFORE CAL JUMP
C
   30 HISUM=0.0
      LOWSUM=0.0
      DO 40 K=1,20
      HISUM=HISUM+DATA(J+K+60)
      LOWSUM=LOWSUM+DATA(J-K-60)
   40 CONTINUE
      FORSLP= (HISUM/20.-LOWSUM/20.)/FLOAT(FLBCAL)
      CAL(1,2)=FORSLP
      ZEROFOR=LOWSUM/20.
      CAL(1,3)=ZEROFOR
C
C      SCALE FORCE CALIBRATION DATA
C
      DO 50 J=1,20*NSECS
      POUNDS(J)=(DATA(J)-ZEROFOR)/FORSLP
   50 CONTINUE
      GO TO 200
C
C      FIND EMG CALIBRATION VARIABLES
C      FIRST REMOVE ANY DC OFFSET IN THE CAL DATA
C
  100 SUM=0.0
      INDEX=1000*NSECS
      DO 110 J=1,INDEX
      SUM=SUM+DATA(J)
  110 CONTINUE
      OFFSET=SUM/FLOAT(INDEX)
      DO 120 J=1,INDEX
      DATA(J)=DATA(J)-OFFSET
      DATA(4000+J)=ABS(DATA(J)-OFFSET)
  120 CONTINUE
```

```
C
C      CALCULATE STANDARD DEVIATION AND OBTAIN SCALE FACTOR. FOR ZERO MEAN,
C      STATIONARY SIGNAL STANDARD DEVIATION=RMS VALUE
C
      XSQUAR=0.0
      DO 140 J=1 ,INDEX
      XSQUAR=XSQUAR+DATA(J)**2
  140 CONTINUE
      EMGCAL=SQRT(XSQUAR/(INDEX-1))/FLOAT(MAGTUD)
      CAL(2,1)=FLOAT(MAGTUD)
      CAL(2,2)=EMGCAL
C
C      SCALE CAL DATA FOR PLOTTING.
C
      DO 180 J=1,INDEX
      DATA(J)=DATA(J)/EMGCAL
  180 CONTINUE
  200 RETURN
      END
      SUBROUTINE GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
C
C      SUBROUTINE GRAPH PLOTS ON THE COMPUTEK TERMINAL THE DATA SLICE BEING
C        PROCESSED ALONG WITH VARIOUS HEADER INFORMATION.
C
C        CALLING ARGUMENTS:  DTYPE - TYPE OF DATA
C
C                            START - BEGIN TIME OF PRESENT DATA SLICE
C
C                            NSECS - LENGTH OF DATA SLICE IN SECONDS
C
C                            SPAN  - AVERAGING INTERVAL FOR PSD
C
C                            STDERR- NORMALIZED STANDARD ERROR FOR PSD ESTIMATE
C
      DIMENSION Z(15),DATE(3),POUNDS(2600),PSDN(410)
      DIMENSION FREQ(410),CALDATA(2000)
      INTEGER DTYPE
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      EQUIVALENCE (POUNDS,DATA(1)),(PSDN,DATA(8000))
      EQUIVALENCE (FREQ,DATA(5000)),(FLBCAL,CAL(1,1))
      EQUIVALENCE (CALDATA,DATA(1)),(EMGMAX,DATA(8500))
      EQUIVALENCE (FORMAX,DATA(8500))
      SUB=FLOAT(IHEAD(1))
      RUN=FLOAT(IHEAD(9))
      DATE(1)=FLOAT(IHEAD(2))
      DATE(2)=FLOAT(IHEAD(3))
      DATE(3)=FLOAT(IHEAD(4))
      FLBCAL=FLOAT(IHEAD(15))
      SHOUR=FLOAT(IFIX(START/3600.))
      SMIN=FLOAT(IFIX((START-SHOUR*3600.)/60.))
      SSEC=START-SHOUR*3600.-SMIN*60.
      FINISH=START+FLOAT(NSECS)
      FHOUR=FLOAT(IFIX(FINISH/3600.))
      FMIN=FLOAT(IFIX((FINISH-FHOUR*3600.)/60.))
      FSEC=FINISH-FHOUR*3600.-FMIN*60.
C
C      INITIALIZE PLOT AND OUTPUT HEADER INFORMATION.
C
      CALL INITAL (Z)
      CALL MODSET(Z,4,10.)
      CALL MODSET(Z,5,10.)
      CALL OBJECT(Z,75.,1008.,100.,770.)
      CALL WORDS(Z,660.,780.,16,'  :  :      :  :')
      CALL WORDS(Z,100.,30.,4,'SUB#')
      CALL WORDS(Z,440.,30.,16,'EXP DATE   /  / ')
      CALL WORDS(Z,840.,30.,4,'RUN#')
      CALL NUMBER(Z,660.,780.,SHOUR,2,0)
      CALL NUMBER(Z,696.,780.,SMIN,2,0)
      CALL NUMBER(Z,732.,780.,SSEC,2,0)
      CALL NUMBER(Z,780.,780.,FHOUR,2,0)
      CALL NUMBER(Z,816.,780.,FMIN,2,0)
      CALL NUMBER(Z,852.,780.,FSEC,2,0)
      CALL NUMBER(Z,160.,30.,SUB,3,0)
      CALL NUMBER(Z,548.,30.,DATE(1),2,0)
      CALL NUMBER(Z,584.,30.,DATE(2),2,0)
      CALL NUMBER(Z,620.,30.,DATE(3),2,0)
      CALL NUMBER(Z,880.,30.,RUN,2,0)
C
C      DECIDE WHERE TO GO FOR EACH DATA TYPE
C
      GO TO (10,40,40,10,70),DTYPE
C
C      LABEL AXES FOR FORCE DATA
C
   10 CALL MODSET(Z,2,1.)
      NPTS=20*NSECS
      PTS=FLOAT(NPTS)
      CALL WORDS(Z,13.,380.,4,'LBS ')
      CALL WORDS(Z,25.,355.,2,'OF')
      CALL WORDS(Z,0.,330.,6,'FORCE ')
      CALL WORDS(Z,450.,55.,16,'TIME IN SECONDS ')
      IF(DTYPE.EQ.4) GO TO 30
C
C      FINISH UP PLOT FOR FORCE CALIBRATION DATA
C
      CALL WORDS(Z,100.,780.,16,'FORCE CALIBRATION')
      CALL WORDS(Z,350.,0.,26,'AMOUNT OF FORCE CAL (LBS) =')
      CALL NUMBER (Z,674.,0.,FLBCAL,3,0)
      CALL SUBJEC (Z,0.,FLOAT(NSECS),0.,200.)
      CALL GRID(Z,2.,25.)
      CALL LABEL(Z,1,2.,2,0)
      CALL LABEL (Z,2,50.,3,0)
      CALL SUBJEC (Z,0.,PTS,0.,200.)
      CALL MODLINE(Z,NPTS,1,POUNDS)
      CALL BELL(Z)
      CALL PAGE(Z)
      GO TO 90
C
C      FINISH UP PLOT FOR FORCE DATA
C
   30 CALL WORDS(Z,100.,780.,10,'FORCE DATA')
      FMAX=200.
      IF(FORMAX.LT.80.) FMAX=100.
      CALL SUBJEC (Z,0.,FLOAT(NSECS),0.,FMAX)
      CALL GRID (Z.2. ,10. )
      TICK=2.+FLOAT(NSECS/31)
      CALL LABEL (Z,1,TICK,3,0)
      CALL LABEL (Z,2,20. ,3,0)
      CALL SUBJEC (Z,0. ,PTS,0. ,FMAX)
      CALL MODLINE(Z,NPTS,1,POUNDS)
      CALL BELL(Z)
      CALL PAGE(Z)
      GO TO 90
C
C      FINISH UP PLOT FOR EMG DATA
C
   40 CALL WORDS(Z,430. ,55. ,20,'TIME IN MILLISECONDS')
      CALL MODSET(Z,2,1.)
      CALL WORDS (Z,0.,440.,6,'MICRO ')
      CALL WORDS (Z,0.,410.,6,'VOLTS ')
      NPTS=1000
      SECMIL=FLOAT(1000*NSECS)
      IF(DTYPE.EQ.3) GO TO 50
C
C      EMG CAL DATA
C
      CALL WORDS(Z,100.,780.,8,'EMG CAL ')
      CALL SUBJEC (Z,0.,SECMIL,-1000.,1000.)
      CALL GRID (Z,250.,250.)
      CALL LABEL(Z,1,500.,4,0)
      CALL LABEL (Z,2,500.,5,0)
      CALL MODLINE(Z,NPTS,NSECS,CALDATA)
      GO TO 60
C
C      EMG DATA
C
   50 CALL WORDS(Z,100.,780.,8,'EMG DATA')
      ULIM=(FLOAT(IFIX(EMGMAX/500.))+1.)*500.
      BLIM=-ULIM
      CALL SUBJEC (Z,0.,SECMIL,BLIM,ULIM)
      HACK=ULIM/4.
      CALL LABEL(Z,1,500.,4,0)
      CALL GRID (Z,250.,HACK)
      WNUM=ULIM/2.
      CALL LABEL (Z,2,WNUM,5,0)
      CALL MODLINE(Z,NPTS,NSECS,DATA)
   60 CALL BELL(Z)
      CALL PAGE(Z)
      GO TO 90
C
C      FINISH UP PLOT FOR PSD DATA
C
   70 CALL WORDS(Z,100.,780.,22,'POWER SPECTRAL DENSITY')
      CALL WORDS(Z,490.,55.,18,'FREQUENCY IN HERTZ')
      CALL WORDS (Z,0.,425.,6,'MICRO ')
      CALL WORDS(Z,0.,405.,6,'VOLTS ')
      CALL WORDS (Z,60.,410.,2,'2 ')
      CALL WORDS (Z,0.0,385.,6,'X1000 ')
      CALL WORDS(Z,100.,0.,16,'BANDWIDTH (HZ) =')
      CALL WORDS (Z,440.,0.,26,'NORMALIZED STANDARD ERROR=')
      CALL NUMBER(Z,232.,0.,SPAN,2,2)
      CALL NUMBER (Z,824.,0.,STDERR,1,3)
      CALL SUBJEC (Z,0.,400.,0.,1.1)
      CALL GRID (Z,25.,.05)
      CALL LABEL (Z,1,25.,3,0)
      CALL LABEL (Z,2,.10001,1,1)
C
C      CREATE X-AXIS FOR PSD VALUES
C
      NPTS=IFIX(400./SPAN)
      DO 80 J=1,NPTS
      FREQ(J)=SPAN*FLOAT(J)
   80 CONTINUE
      CALL LINES(Z,NPTS,FREQ,PSDN)
      CALL BELL(Z)
      CALL PAGE(Z)
   90 RETURN
      END
      SUBROUTINE MODLINE (Z,NPTS,MODX,Y)
C
```

```
C       SUBROUTINE MODLINE PLOTS A Y-ARRAY OF DATA POINTS WITHOUT REQUIRING A
C         CORRESPONDING X-ARRAY. INSTEAD OF AN X-ARRAY,THE ARRAY POSITION OF A
C         Y-ELEMENT IS USED FOR THE X VALUE.
C
C         CALLING ARGUMENTS:  Z    - THE PLOT ARRAY
C
C                             NPTS - THE NUMBER OF POINTS TO BE PLOTTED
C
C                             MODX - X VALUE INCREMENT (X=1,EVERY Y POINT; X=2,
C                                    EVERY OTHER Y POINT; ETC.)
C
C                             Y    - THE Y-ARRAY
C
C         SPECIAL CONSIDERATION: NPTS*MODX SHOULD NOT BE GREATER THAN THE LENGTH
C                                OF THE Y-ARRAY
C
      DIMENSION Z(15),Y(1)
      CALL POINTS (Z,1,MODX,Y)
      J2=IWRDXY (Z(2),1)
      DO 70 I=2,NPTS
      IY=I*MODX
      XX=FLOAT (I*MODX)
      GO TO (40,50),J2
   40 YY=WORDXY(Y,IY)
      GO TO 60
   50 YY=Y(IY)
   60 CALL VCTOR(SCALE(Z,XX,0),SCALE(Z,YY,1),1)
   70 CONTINUE
      RETURN
      END
      SUBROUTINE WORDS (Z,X,Y,NL,L)
      DIMENSION Z(15),L(50),M(50)
      NW=NL/2
      DO 10 I=1,NW
   10 M(I)=L(I)
      CALL PASTOR (0,M,NW)
      CALL LEGEND (Z,X,Y,M,NL)
      RETURN
      END
      SUBROUTINE BELL (Z)
      CALL MODE (Z,0)
      CALL OUT (7,1)
      CALL SCRIBE
      DO 10 I=1,300 .
      DO 10 J=1,60
   10 K=I+J
      CALL OUT (Z,1)
      CALL SCRIBE
   20 CALL IN (I)
      IF(I.NE.13) GO TO 20
      RETURN
      END
      SUBROUTINE EMG (NSECS,VOLTSEC)
C
C      SUBROUTINE EMG SCALES EMG DATA FOR PLOTTING ON THE COMPUTER AND FINDS
C        THE INTEGRATED EMG VALUE FOR THE DATA SLICE.
C
C        CALLING ARGUMENTS:   NSECS - LENGTH OF DATA SLICE
C
C                             VOLTSEC- INTEGRATED EMG VALUE IN MICROVOLT*SEC.
C
      INTEGER DTYPE
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      EQUIVALENCE (IRATE,IHEAD(11)),(EMGCAL,CAL(2,2))
      EQUIVALENCE (EMGMAX,DATA(8500)),(EMGVAR,DATA(8499))
      SUM=0.0
C
C        CALCULATE OFFSET, SUBTRACT FROM DATA AND APPLY SCALE FACTOR
C
      DO 10 J=1,1024*NSECS
      SUM=SUM+DATA(J)
   10 CONTINUE
      OFFSET=SUM/(1024.*FLOAT(NSECS))
      CAL(2,4)=OFFSET
      EMGMAX=0.0
      DO 20 J=1,1024*NSECS
```

```
C
      CALL RFORT (DATA,NEXP,S,-1,IFERR)
C
C      CALCULATE RAW POWER SPECTRAL DENSITY
C
      H=1./FLOAT(IRATE)
      K=1
      DO 10 J=3,N+1,2
      PSD(K)=8.*H/PTS*(DATA(J)**2+DATA(J+1)**2)
      K=K+1
   10 CONTINUE
C
C      CORRECT PSD AMPLITUDES FOR WINDOW REDUCTION
C
      DO 15 J=1,N/2
      PSD(J)=1.1429*PSD(J)
   15 CONTINUE
C
C      OUTPUT RAW PSD TO TAPE
C
      CALL PSDSAVE (NSECS)
C
C      CALCULATE SMOOTHED PSD
C
      F0=1./(PTS/FLOAT(IRATE))
      IOTA=IFIX(SPAN/F0)
      SPAN=FLOAT(IOTA)*F0
      PSDMAX=0.0
      K=1
      DO 30 J=1,N/2,IOTA
      TEMP=0.0
      DO 20 JJ=1,IOTA
      TEMP=TEMP+PSD(J+JJ-1)
   20 CONTINUE
      PSD(K)=TEMP/FLOAT(IOTA)
      IF(FLOAT(J/IOTA)*SPAN.GT.400.+SPAN) GO TO 25
      IF(PSD(K).GT.PSDMAX) PSDMAX=PSD(K)
   25 K=K+1
   30 CONTINUE
C
C       CALCULATE THE NORMALIZED PSD
C
      LIM=IFIX(400./SPAN)+1
      DO 40 J=1,LIM
      PSDN(J)=PSD(J)/PSDMAX
   40 CONTINUE
C
C       INTEGRATE THE SMOOTHED PSD
C
      PSDSUM=0.0
      DO 50 J=1,LIM
      PSDSUM=PSDSUM+SPAN*PSD(J)
   50 CONTINUE
C
C       CALCULATE % EACH BANDWIDTH IS OF TOTAL AND FIND THE CUMLATIVE % OF TOTAL
C
      DO 60 J=1,LIM
      PERCNT(J)=SPAN*PSD(J)/PSDSUM*100.
      IF(J.NE.1) GO TO 55
      CUMPCT(1)=PERCNT(1)
      GO TO 60
   55 CUMPCT(J)=CUMPCT(J-1)+PERCNT(J)
   60 CONTINUE
C
C       CALCULATE THE EXPECTED VALUE AND VARIANCE OF THE PSD
C
      F=SPAN/2.
      EXPVAL=0.0
      DO 70 J=1,LIM
      EXPVAL=EXPVAL+(PERCNT(J)/100.)*F
      F=F+SPAN
   70 CONTINUE
      F=SPAN/2.
      VARVAL=0.0
      DO 80 J=1,LIM
      VARVAL=VARVAL+(PERCNT(J)/100.)*(F-EXPVAL)**2
      F=F+SPAN
   80 CONTINUE
      STDEV=SQRT(VARVAL)
C
C      FIND STANDARDIZED NORMAL ERROR
C
      STDERR=SQRT(1./FLOAT(IOTA))
      RETURN
      END
      SUBROUTINE WINDOW (NSECS)
C
C      SUBROUTINE WINDOW APPLIES A COSINE TAPER TO THE FIRST AND LAST TENTHS OF
C        THE DATA TO REDUCE LEAKAGE.
C
C        CALLING ARGUMENTS:  NSECS - LENGTH OF THE DATA
C
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      PI=3.1415927
      NPTS=1024*NSECS
      IEDGE=NPTS/10
      TTOTAL=1.024*FLOAT(NSECS)
      K=1
      DO 10 J=1,IEDGE
      T1=.001*FLOAT(J)
      C1=.5*(1.-COS(PI*T1/(.1*TTOTAL)))
      DATA(J)=C1*DATA(J)
      DATA(NPTS-J+1)=C1*DATA(NPTS-J+1)
   10 CONTINUE
      RETURN
      END
      SUBROUTINE RFORT(A,M,S,IFS,IFERR)
C     ONE-DIMENSIONAL REAL FINITE FOURIER TRANSFORM
C
C
C     FOURIER TRANSFORM SUBROUTINE FOR REAL DATA.
C     PROGRAMMED IN SYSTEM/360, BASIC PROGRAMMING SUPPORT,
C     FORTRAN IV, (SEE FORM  C28-6504).
C     THIS DECK IS SET UP FOR IBSYS ON THE IBM 7094
```

```
C
C       THIS PROGRAM USES THE SUBROUTINE FORT TO COMPUTE COMPLEX
C       FOURIER TRANSFORMS OF REAL DATA.  PK FORT  S.D.A. NO. 3465 IS
C       AVAILABLE THROUGH SHARE.
C
C       THE FOURIER SERIES IS
C
C       X(J)= SUM OVER K=0 TO N, OF C(K)*EXP(2*PI*I*J*K/N )
C       J=0,1,2,...,N-1
C
C       WHERE I=SQRT(-1) AND WHERE C(K) IS COMPLEX.
C       SINCE X(J) IS REAL, C(K)= CONJG(C(N-K) ). THEREFORE ONLY
C       C(K),K= 0,1,...,N/2 ARE COMPUTED AND/OR USED.
C
C       ARGUMENTS-
C
C       A IS INITIALLY THE INPUT ARRAY, X, WHEN COMPUTING A FOURIER
C       TRANSFORM AND C WHEN COMPUTING A FOURIER SERIES. A IS REPLACES BY
C       THE OUTPUT ARRAY, C IN THE FORMER CASE, X IN THE LATTER.
C       THE X VECTOR CONTAINS THE REAL DATA X(0),X(1),...,X(N-1)
C       THE C VECTOR CONTAINS THE COMPLEX FOURIER AMPLITUDES
C       C(0),C(1),...,C(N/2). THE COMPLEX VECTOR C IS STORED ACCORDING
C       TO THE NORMAL FORTRAN IV CONVENTION FOR STORING COMPLEX NUMBERS.
C       I.E., REAL PARTS IN ALTERNATE CELLS STARTING WITH THE FIRST,
C       IMAGINARY PARTS IN ALTERNATE CELLS STARTING WITH THE SECOND.
C       TO ADHERE TO FORTRAN RULES, X(0), X(1),..., ARE REFERRED TO AS
C       X(1),X(2),..., RESP. IN THE PROGRAMS.  ALSO, C(0),C(1),... ARE
C       REFERRED TO AS C(1),C(2),..., RESP., IF C IS DESIGNATED AS
C       COMPLEX IN A TYPE STATEMENT.
C
C       M GIVES N=2**M
C
C       THE ARGUMENTS S, IFS, AND IFERR ARE THE SAME AS IN THE
C       SUBROUTINE FORT AND THE USER IS REFERRED TO THE COMMENT CARDS
C       IN FORT THEIR EXPLANATION.
C       DIMENSION STATEMENTS- THE DIMENSIONS OF ARRAYS A AND S SHOULD
C       BE N+2 AND N/4, RESP. FOR THE LARGEST N TO BE USED. FOR
C       EXAMPLE, IF THE LARGEST M IS 13, THEN, N=8192 AND ONE SHOULD
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C      HAVE THE DIMENSION STATEMENT-
C      DIMENSION A(8194), S(2048)
C      IF ONE WISHES TO SPECIFY A TO BE COMPLEX BY A TYPE STATEMENT,
C      ONE SHOULD GIVE IT A DIMENSION OF N/2 +1 , FOR THE LARGEST N
       DIMENSION A(8500),S(1024)
   10  IFERRS = 0
       N=2**M
       NV2 = N / 2
       NV4M1 = N/4 - 1
       MM1 = M - 1
       IF (ABS(IFS)-1) 40,40,20
   20  IF (MP-M) 30,50,50
   30  IFERRS = 1
   40  NP = N
       MP = M
       CALL FORT(A,M,S,0,IFERR1)
       IFERRS = IFERRS + IFERR1
   50  KD = NP / N
       KT = KD
       NPV4 = NP / 4
       IF (IFS) 60,80,90
   60  CALL FORT(A,MM1,S,-2,IFERR2)
       IFERRS = IFERRS + IFERR2
       DO 70 K=1,NV4M1
       J=NV2-K
       A1R= A(2*K+1) + A(2*J+1)
       A1I=A(2*K+2)-A(2*J+2)
       A2R=A(2*K+2)+A(2*J+2)
       A2I=A(2*J+1)-A(2*K+1)
       KKT = NPV4-KT
       AWR=A2R*S(KKT) + A2I*S(KT)
       AWI = A2I*S(KKT)- A2R*S(KT)
       A(2*K+1)=(A1R+AWR)/4.
       A(2*K+2)=(A1I+AWI)/4.
       A(2*J+1)=(A1R-AWR)/4.
       A(2*J+2)=(AWI-A1I)/4.
   70  KT=KT+KD
       T=A(1)
```

```
      A(1)=(T+A(2))/2
      A(N+1) = (T-A(2))/2
      A(2)=0
      A(N+2) = 0
      A(NV2+1) =  .5*A(NV2+1)
      A(NV2+2) = -.5 * A(NV2+2)
   80 IFERR = IFERRS
      RETURN
   90 DO 100 K=1,NV4M1
      J=NV2-K
      A1R=A(2*K+1) + A(2*J+1)
      A1I=A(2*K+2)-A(2*J+2)
      AWR=A(2*K+1)-A(2*J+1)
      AWI=A(2*K+2)+A(2*J+2)
      KKT = NPV4 - KT
      A2R=AWR*S(KKT) - AWI*S(KT)
      A2I=AWR*S(KT) + AWI*S(KKT)
      A(2*K+1) = A1R - A2I
      A(2*K+2) = A1I + A2R
      A(2*J+1) = A1R + A2I
      A(2*J+2) = A2R - A1I
  100 KT = KT + KD
      T = A(1)
      A(1) = T + A(N+1)
      A(2) = T - A(N+1)
      A(NV2+1) = 2.*A(NV2+1)
      A(NV2+2) =-2.*A(NV2+2)
      CALL FORT(A,MM1,S,2,IFERR2)
      IFERRS = IFERRS+IFERR2
      GO TO 80
      END
      SUBROUTINE FORT(A,M,S,IFS,IFERR)
C           FORT, ONE-DIMENSIONAL FINITE COMPLEX FOURIER TRANSFORM.
C
C
C     FOURIER TRANSFORM SUBROUTINE, PROGRAMMED IN SYSTEM/360,
C     BASIC PROGRAMMING SUPPORT, FORTRAN IV, FORM C28-6504
C     THIS DECK SET UP FOR IBSYS ON IBM 7094.
```

```
C
C     DOES EITHER FOURIER SYNTHESIS,I.E.,COMPUTES COMPLEX FOURIER SERIES
C     GIVEN A VECTOR OF N COMPLEX FOURIER AMPLITUDES,OR, GIVEN A VECTOR
C     OF COMPLEX DATA X DOES FOURIER ANALYSIS, COMPUTING AMPLITUDES
C     A IS A COMPLEX VECTOR OF LENGTH N=2**M COMPLEX NOS. OR 2*N REAL
C     NUMBERS. A IS TO BE SET BY USER
C     M  IS AN INTEGER 0.LT.M.LE.13, SET BY USER.
C     S IS A VECTOR S(J)= SIN(2*PI*J/NP ), J=1,2,....,NP/4-1,
C     COMPUTED BY PROGRAM.
C     IFS IS A PARAMETER TO BE SET BY USER AS FOLLOWS-
C     IFS=0 TO SET NP=2**M AND SET UP SINE TABLE.
C     IFS=1 TO SET N=NP=2**M, SET UP SIN TABLE, AND DO FOURIER
C     SYNTHESIS, REPLACING THE VECTOR A BY
C
C     X(J)= SUM OVER K=0,N-1 OF A(K)*EXP(2*PI*I/N)**(J*K),
C     J=0,N-1, WHERE I=SQRT(-1)
C
C     THE  X'S  ARE STORED WITH   RE X(J) IN CELL  2*J+1
C     AND  IM X(J)  IN CELL  2*J+2  FOR  J=0,1,2,...,N-1.
C     THE  A'S  ARE STORED IN THE SAME MANNER.
C
C     IFS=-1   TO SET  N=NP=2**M,SET UP SIN TABLE, AND DO FOURIER
C     ANALYSIS, TAKING THE INPUT VECTOR A AS X AND
C     REPLACING IT BY THE A  SATISFYING THE ABOVE FOURIER SERIES.
C     IFS>0   SET UP SIN TABLE AND RETURN
C     IFS=+2 TO DO FOURIER SYNTHESIS ONLY, WITH A PRE-COMPUTED S.
C     IFS=-2 TO DO FOURIER ANALYSIS ONLY, WITH A PRE-COMPUTED S.
C
C     IFERR   IS SET BY PROGRAM TO-
C     =0 IF NO ERROR DETECTED.
C     =1 IF M IS OUT OF RANGE., OR, WHEN IFS=+2,-2, THE
C     PRE-COMPUTED S TABLE IS NOT LARGE ENOUGH.
C     =-1 WHEN IFS =+1,-1, MEANS ONE IS RECOMPUTING S TABLE
C     UNNECESSARILY.
C
C     NOTE- AS STATED ABOVE, THE MAXIMUM VALUE OF M FOR THIS PROGRAM
C     ON THE IBM 7094 IS 13. FOR 360 MACHINES HAVING GREATER STORAGE
C     CAPACITY, ONE MAY INCREASE THIS LIMIT BY REPLACING
```

```
C      STATEMENT 3 BELOW BY LOG2 N, WHERE N IS THE MAX  NO. OF
C      COMPLEX NUMBERS ONE CAN STORE IN HIGH-SPEED CORE.  ONE MUST
C      ALSO ADD MORE  DO  STATEMENTS TO THE BINARY SORT ROUTINE
C      FOLLOWING STATEMENT  24  AND CHANGE THE EQUIVALENCE STATEMENTS
C      FOR THE  K'S.
C
       DIMENSION A(8500),S(1024),K(15)
       IF (M) 20,20,10
    10 IF (M-15) 40,40,20
    20 IFERR=1
    30 RETURN
    40 IFERR=0
       N=2**M
       IF (IABS(IFS)-1) 440,440,50
C      WE ARE DOING TRANSFORM ONLY. SEE IF PRE-COMPUTED
C      S TABLE IS SUFFICIENTLY LARGE
    50 IF (N-NP) 70,70,60
    60 IFERR=1
       GO TO 440
C      SCRAMBLE A, BY SANDE'S METHOD
    70 K(1)=2*N
       DO 80 L=2,M
    80 K(L)=K(L-1)/2
       DO 90 L=M,14
    90 K(L+1)=2
C       THE FOLLOWING 15 STATEMENTS ARE TO COMPENSATE FOR A WEAKNESS IN
C      THE FORTRAN V COMPILER
       K1 =K(15)
       K2 =K(14)
       K3 =K(13)
       K4 =K(12)
       K5 =K(11)
       K6 =K(10)
       K7 =K(9)
       K8 =K(8)
       K9 =K(7)
       K10=K(6)
       K11=K(5)
```

```
      K12=K(4)
      K13=K(3)
      K14=K(2)
      K15=K(1)
      N2 =K(1)
C     NOTE EQUIVALENCE OF KL AND K(14-L)
C     BINARY SORT-
      IJ =2
  100 J1 =2
  110 J2 =J1
  120 J3 =J2
  130 J4 =J3
  140 J5 =J4
  150 J6 =J5
  160 J7 =J6
  170 J8 =J7
  180 J9 =J8
  190 J10=J9
  200 J11=J10
  210 J12=J11
  220 J13=J12
  230 J14=J13
  240 JI=J14
  250 IF (IJ-JI) 260,270,270
  260 T=A(IJ-1 )
      A(IJ-1)=A(JI-1)
      A(JI-1)=T
      T=A(IJ)
      A(IJ)=A(JI)
      A(JI)=T
  270 IJ=IJ+2
      JI=JI+K14
      IF (JI.LE.K15) GO TO 250
      J14=J14+K13
      IF (J14.LE.K14) GO TO 240
      J13=J13+K12
      IF (J13.LE.K13) GO TO 230
      J12=J12+K11
      IF (J12.LE.K12) GO TO 220
      J11=J11+K10
      IF (J11.LE.K11) GO TO 210
      J10=J10+K9
      IF (J10.LE.K10) GO TO 200
      J9 =J9+K8
      IF (J9.LE.K9) GO TO 190
      J8=J8+K7
      IF (J8.LE.K8) GO TO 180
      J7=J7+K6
      IF (J7.LE.K7) GO TO 170
      J6=J6+K5
      IF (J6.LE.K6) GO TO 160
      J5=J5+K4
      IF (J5.LE.K5) GO TO 150
      J4=J4+K3
      IF (J4.LE.K4) GO TO 140
      J3=J3+K2
      IF (J3.LE.K3) GO TO 130
      J2=J2+K1
      IF (J2.LE.K2) GO TO 120
      J1=J1+2
      IF (J1.LE.K1) GO TO 110
      IF (IFS) 280,20,300
C     DOING FOURIER ANALYSIS,SO DIV. BY N AND CONJUGATE.
  280 FN = N
      DO 290 I=1,N
      A(2*I-1)=A(2*I-1)
  290 A(2*I)=-A(2*I)
C     SPECIAL CASE- L=1
  300 DO 310 I=1,N,2
      T = A(2*I-1)
      A(2*I-1) =T + A(2*I+1)
      A(2*I+1)=T-A(2*I+1)
      T=A(2*I)
      A(2*I) = T + A(2*I+2)
  310 A(2*I+2)= T - A(2*I+2)
      IF (M-1) 20,30,320
```

```
C      SET FOR L=2
  320 LEXP1=2
C      LEXP1=2**(L-1)
       LEXP=8
C      LEXP=2**(L+1)
       NPL= 2**MT
C      NPL = NP* 2**-L
  330 DO 390 L=2,M
C      SPECIAL CASE- J=0
       DO 340 I=2,N2,LEXP
       I1=I + LEXP1
       I2=I1+ LEXP1
       I3 =I2+LEXP1
       T=A(I-1)
       A(I-1) = T +A(I2-1)
       A(I2-1) = T-A(I2-1)
       T =A(I)
       A(I) = T+A(I2)
       A(I2) = T-A(I2)
       T= -A(I3)
       TI = A(I3-1)
       A(I3-1) = A(I1-1) - T
       A(I3   ) = A(I1 )    - TI
       A(I1-1) = A(I1-1) +T
  340 A(I1)    = A(I1    )  +TI
       IF (L-2) 380,380,350
  350 KLAST=N2-LEXP
       JJ=NPL
       DO 370 J=4,LEXP1,2
       NPJJ=NT-JJ
       UR=S(NPJJ)
       UI=S(JJ)
       ILAST=J+KLAST
       DO 360 I=J,ILAST,LEXP
       I1=I+LEXP1
       I2=I1+LEXP1
       I3=I2+LEXP1
       T=A(I2-1)*UR-A(I2)*UI
```

```
      IARRAY(J)=0
   10 CONTINUE
      DO 20 J=1,20
      IARRAY(J)=IHEAD(J)
   20 CONTINUE
      NRECS=NSECS/2
      IF(NRECS.EQ.0) NRECS=1
      IARRAY(21)=NRECS
      CALL QWRITE(IOUTP,25,INDIC)
C
C     OUTPUT FOLLOWING DATA RECORDS CONTAINING THE RAW PSD VALUES
C
      K=0
      DO 50 J=1,NRECS
      DO 40 JJ=1,1024
      DATA(4099+JJ)=DATA(K+JJ)
   40 CONTINUE
      CALL QWRITE (IOUTP,2048,INDIC)
      K=K+1024
   50 CONTINUE
C
C      WRITE TWO EOFS AND BACK UP TO JUST BEFORE THEM
C
      CALL QWEOF (IOUTP,1,INDIC)
      CALL QCLOSE (IOUTP,0)
      RETURN
      END
      SUBROUTINE PRINT (DTYPE,START,NSECS,SPAN,VOLTSEC,PSDSUM,STDERR,WHO
     @A)
C
C      SUBROUTINE PRINT OUTPUTS TO THE LINE PRINTER HEADER AND CALIBRATION DATA,
C       AVERAGED FORCE DATA, AND SMOOTHED PSD SPECTRUM.
C
C       CALLING ARGUMENTS    DTYPE - TYPE OF DATA
C
C                            START - BEGIN TIME OF PRESENT DATA SLICE
C
C                            NSECS - LENGTH OF DATA SLICE
```

```
C
C                    SPAN   - FOR FORCE DATA NUMBER OF SECS FORCE DATA IS
C                             AVERAGED OVER; FOR PSD DATA FREQUENCY
C                             INTERVAL FOR SMOOTHING ESTIMATES
C
C                    VOLTSEC- INTEGRATED EMG VALUE (TIME DOMAIN)
C
C                    PSDSUM- INTEGRATED EMG VALUE (FREQUENCY DOMAIN)
C
C                    STDERR- NORMALIZED STANDARD ERROR OF ESTIMATE FOR
C                            PSD VALUES
C
C                    WHOA   - FLAG SET WHEN DATA PROCESSING ATTEMPTED
C                             WITHOUT CALIBRATION
C
C
      INTEGER DTYPE,WHOA
      DIMENSION IEDATE(3),IDDATE(3),MUSCLE(7,4),IFORR(2)
      DIMENSION AVGLBS(130),PSD(410),ITIME(3),PSDN(410),PERCNT(410)
      DIMENSION CUMPCT(410)
      COMMON DATA(8500),IHEAD(20),CAL(2,4)
      EQUIVALENCE (ISUBNO,IHEAD(1)),(IEDATE,IHEAD(2))
      EQUIVALENCE (IDDATE,IHEAD(5)),(IFLITE,IHEAD(8))
      EQUIVALENCE (IRUN,IHEAD(9)),(MUS,IHEAD(20))
      EQUIVALENCE (IATAPE,IHEAD(10)),(ISAMPE,IHEAD(11))
      EQUIVALENCE (ISAMPM,IHEAD(12)),(AVGLBS,DATA(5000))
      EQUIVALENCE (PSD,DATA(1)),(PSDN,DATA(8000))
      EQUIVALENCE (PERCNT,DATA(7000)),(CUMPCT,DATA(7500))
      EQUIVALENCE (EXPVAL,DATA(6500)),(STDEV,DATA(6501))
      EQUIVALENCE (EMGVAR,DATA(8499))
      DATA MUSCLE /'BR','AC','HI','AL',' B','IC','EP',' B','  ','RA','DI
     @','AL','IS','  ',' G','AS','TR','OC','NE','MI','US',' S','OL','EU'
     @,'S ','  ','  ','  '/
      DATA IFORR /' F','R+'/
      IF(WHOA.EQ.1) GO TO 150
      ITIME(1)=IFIX(START/3600.)
      ITIME(2)=IFIX((START-FLOAT(ITIME(1))*3600.)/60.)
      ITIME(3)=IFIX(START-FLOAT(ITIME(1))*3600.-FLOAT(ITIME(2))*60.)
```

```
C
C      BRANCH TO PROPER PRINT OUT SECTION FOR THE DATA TYPE
C
      IF(DTYPE-4) 5,50,100
C
C      PRINT OUT HEADER
C
    5 IP=1
      IF (IFLITE.GE.0) IP=2
      WRITE(108,10)
   10 FORMAT(1H1,//,48X,'CARDIOVASCULAR LABORATORY',//,47X,'EMG DATA PRO
     @CESSING PROGRAM',/,47X,'---------------------------')
      WRITE(108,20)ISUBNO,IEDATE,IFORR(IP),IFLITE,IRUN,IATAPE,IDDATE,(MU
     @SCLE(J,MUS),J=1,7),ISAMPE,ISAMPM
   20 FORMAT(////,47X,'*** HEADER INFORMATION***',///,' SUBJECT NO.:', 7
     @X,I3,18X,'EXPERIMENT DATE:',6X,I2,'/',I2,'/',I2,10X,'FLIGHT REFERE
     @NCE DAY:',A3,I3,//,' RUN NO.:',11X,I2,19X,'ANALOG TAPE NO.:', 7X,I
     @3,14X,'DIGITIZING DATE:',6X,I2,'/',I2,'/',I2,//,' MUSCLE:', 9X,7A2
     @,//,' EMG SAMPLE RATE (SAMP/SEC):',I5,8X,'FORCE SAMPLE RATE (SAMP/
     @SEC):',I5)
      WRITE(108,30)(CAL(1,J),J=1,4)
   30 FORMAT(////,49X,'** CALIBRATION DATA **',///,' FORCE CAL',/,' ----
     @-----',//,' CAL WEIGHT (LBS):',F7.1,16X,'COUNTS/LB:',2X,F6.0,5X,'A
     @VG. BASELINE COUNT:',2X,F6.0,5X,'CAL GAIN:',2X,F3.0)
      WRITE(108,40) (CAL(2,J),J=1,2)
   40 FORMAT(///,' EMG CAL',/,' -------',//,' RMS AMPLITUDE (MICROVOLTS)
     @:',F6.0,7X,' COUNTS/MICROVOLT:',F5.2,//)
      GO TO 200
C
C      PRINT OUT FORCE DATA
C
   50 WRITE(108,60) ITIME,NSECS,SPAN
   60 FORMAT(1H1,51X,'*** FORCE DATA ***'///5X,'START TIME-',I3,':',I2,'
     @:',I2,15X,'DATA LENGTH (SECS)-',I4,15X,'AVERAGE FORCE INTERVAL (SE
     @CS)-',F3.0,//)
      LOOPS=NSECS/IFIX(SPAN)
      IF(LOOPS.GT.49) GO TO 66
      WRITE(108,62)
   62 FORMAT(47X,'INTERVAL',5X,'AVERAGE FORCE (LBS)'/,47X,'--------',5X
     @'-------------------',//)
      DO 65 J=1,LOOPS
      WRITE(108,64) J,AVGLBS(J)
   64 FORMAT(49X,I3,15X,F5.1)
   65 CONTINUE
      GO TO 200
C
C      PRINT OUT PSD DATA
C
   66 IF((LOOPS/2)*2.NE.LOOPS) LOOPS=LOOPS-1
      WRITE(108 ,68)
   68 FORMAT(15X,'INTERVAL',5X,'AVERAGE FORCE (LBS)',20X,'INTERVAL',5X,'
     @AVERAGE FORCE (LBS)',/,15X,'--------',5X,'-------------------',20X
     @,'--------',5X,'-------------------',//)
      DO 75 J=1,LOOPS/2
      JJ=J+LOOPS/2
      WRITE(108,73) J,AVGLBS(J),JJ,AVGLBS(JJ)
   73 FORMAT(17X,I3,15X,F5.1,29X,I3,15X,F5.1)
   75 CONTINUE
      GO TO 200
  100 WRITE(108,110) ITIME,NSECS,VOLTSEC,SPAN,STDERR,PSDSUM,EXPVAL,STDEV
     @,EMGVAR
  110 FORMAT(1H1,39X,'*** POWER SPECTRAL DENSITY OF EMG DATA ***'///4X,'
     @START TIME-',I3,':',I2,':',I2,11X,'DATA LENGTH (SECS)-',I4,14X,'IN
     @TEGRATED EMG (MICROVOLT*SEC)-',E12.4//4X,'BANDWIDTH (HZ)-',F6.3,10
     @X,'NORMALIZED STANDARD ERROR-',F6.3,5X,'INTEGRATED PSD (MICROVOLT*
     @*2)-'      ,E13.4,//,4X,'MEAN (HZ)-',F6.1,15X,'STANDARD DEVIATION
     @(HZ)-',F6.1,8X,'EMG VARIANCE (MICROVOLT**2)-',E15.4,/)
      LOOPS=IFIX(400./SPAN)+1
      FREQ=SPAN/2
      IF(LOOPS.GT.46) GO TO 130
      WRITE(108,115)
  115 FORMAT(30X,' FREQ',13X,'PSD',11X,'PSD',10X,'% OF',10X,'CUM %'/,30X
     @,' (HZ)',8X,'(MMV**2/HZ)',8X,'NORM',9X,'TOTAL',9X,'TOTAL',/, 31X,'
     @----',8X,'-----------' ,8X,'----',9X,'-----',9X,'-----',/)
      DO 125 J=1,LOOPS
      WRITE(108,120) FREQ,PSD(J),PSDN(J),PERCNT(J),CUMPCT(J)
```

```
 120 FORMAT(30X,F6.2,7X,G10.5,8X,F6.4,7X,F6.2,8X,F6.2)
     FREQ=FREQ+SPAN
 125 CONTINUE
     GO TO 200
 130 IF((LOOPS/2)*2.NE.LOOPS) LOOPS=LOOPS-1
      FREQ2=FLOAT(LOOPS/2)*SPAN+FREQ
     WRITE(108,135)
 135 FORMAT('  FREQ', 8X,'PSD', 9X,'PSD',8X,'% OF',6X,'CUM %',22X,'FREQ
    @', 9X,'PSD' , 8X,'PSD',8X,'% OF',6X,'CUM %'/2X,'(HZ)',4X,'(MMV**2/
    @HZ)',5X,'NORM',7X,'TOTAL',5X,'TOTAL',22X,'(HZ)',5X,'(MMV**2/HZ)',5
    @X,'NORM',6X,'TOTAL',6X,'TOTAL',/,'  ----',4X,'-----------',5X,'---
    @-',7X,'-----',5X,'-----',22X,'----',5X,'-----------',5X,'----',6X,
    A'-----' ,6X,'-----',/)
     DO 145 J=1,LOOPS/2
     JJ=J+LOOPS/2
     WRITE(108,140) FREQ,PSD(J),PSDN(J),PERCNT(J),CUMPCT(J),FREQ2,PSD(J
    @J),PSDN(JJ),PERCNT(JJ),CUMPCT(JJ)
 140 FORMAT(F7.2,3X,G10.5,5X,F6.4,5X,F6.2,5X,F6.2,20X,F6.2,4X,G10.5,5X,
    @F6.4,5X,F6.2,5X,F6.2)
     FREQ=FREQ+SPAN
     FREQ2=FREQ2+SPAN
 145 CONTINUE
     GO TO 200
C
C     ERROR MESSAGE
C
 150 WRITE(108,160)
 160 FORMAT(' ERROR...DATA ANALYSIS ATTEMPTED WITHOUT CALIBRAT\-NN \IsST
    @    CHECK DATA CARD ORDER.')
     STOP
 200 RETURN
     END
R
```

```
!JOB  EH

!ASS GO=EMGGO,UD

!ASS OV=EMGAN,UP

!OLOAD 7,B
 !$ROOT 400,,GO,1
 !$SEG 1,0,GO,1
 !$SEG 2,0,GO,3
 !$SEG 3,0,GO,1
 !$SEG 4,0,GO,4
 !$SEG 5,0,GO,2
 !$SEG 6,0,GO,5
 !$SEG 7,0,GO,1
 !$ML
EOD
```

```
MAP

OVERLAY TASK  BA  ØRG=2100 HLØC=3AD4 CBAS=3574 CSIZ=A28C UPEN=029F

          DEF    M:FSAVE                0183
          DEF    D:KEY                  1758
          DEF    D:CARD                 1759
          DEF    D:SNAP                 175A
          DEF    M:SAVE                 1753
          DEF    M:EXIT                 1754
          DEF    M:IØEX                 175B
          DEF    M:READ                 175D
          DEF    M:WRITE                175E
          DEF    M:CTRL                 175F
          DEF    M:TERM                 1761
          DEF    M:DATIME               1760
          DEF    M:ABØRT                1762
          DEF    M:HEXIN                1763
          DEF    M:INHEX                1764
          DEF    M:CKREST               1765
          DEF    M:LØAD                 1755
          DEF    M:ØPEN                 1766
          DEF    M:CLØSE                1767
          DEF    M:DKEYS                1768
          DEF    M:WAIT                 1769
          DEF    M:SEGLD                176A
          DEF    M:DEFINE               176B
          DEF    M:ASSIGN               176C
          DEF    M:ØPFILE               176D
          DEF    M:PØP                  176E
          DEF    M:RES                  176F
          DEF    M:DYN                  1770
          DEF    M:RSVP                 1756
          DEF    M:DØW                  1757
          DEF    M:CØC                  175C

RØØT  ØRG=2290  LWA=264A  LEN=03BB  TRA=2290  SEV=0000  ØV:LØAD=2415

          DEF    L:88Z      L  S  M     263F
          DEF    M:PSHC     L  S  M     25EC
          DEF    M:PUSH     L  S  M     25EF
          DEF    L:ERRØR    L  S  M     257C
          DEF    SEGLDX     L  S  M     2546
          DEF    SEGLD      L  S  M     2544
          DEF    L:88X1     L  S  M     24A9
          DEF    L:88X      L  S  M     24A7
          DEF    L:33R3     L  S  M     248F
          DEF    L:33R2     L  S  M     2493
          DEF    L:33R1     L  S  M     2497
          DEF    M:SR       L  S  M     2630
          DEF    L:32R4     L  S  M     2444
          DEF    L:32L4     L  S  M     247D
          DEF    L:32L3     L  S  M     2477
          DEF    L:32L2     L  S  M     2473
          DEF    L:32L1     L  S  M     247B
          DEF    L:32R3     L  S  M     243E
          DEF    L:32R2     L  S  M     243A
          DEF    L:32R1     L  S  M     2442
          DEF    L:32C      L  S  M     2483
          DEF    ::FLOAT     L S M        2480
          DEF    :FLOAT      L S M        2480
          DEF    OV:LOAD     I            2415
       P  REF    PRINT       I            0007
       P  REF    FORCE       I            0005
       P  REF    FFTPSD      I            0006
       P  REF    FMG    .    I            0005
       P  REF    GRAPH       I            0004
       P  REF    DCAL        I            0003
       P  REF    OUT60       I            0002
       P  REF    TINPUT      I            0002
       P  REF    CINPUT      I            0001
          DEF    F:MAIN      I            2290

SEGMENT IDENT NODE  ORG   LWA   LEN   TRA   SEV
         0001 0000  264B  3610  0FC6  0000  0000

          DEF    L:3N        L S M        35FE
          DEF    L:44R2      L S M        35E4
          DEF    L:44R1      L S M        35E8
          DEF    L:44M2      L S M        3527
          DEF    L:44M1      L S M        352F
          DEF    L:44L4      L S M        3517
          DEF    L:44L1      L S M        3515
          DEF    L:44T4      L S M        34EE
          DEF    L:44T2      L S M        34E4
          DEF    L:44T1      L S M        34EC
          DEF    X:3NORM     L S M        35A4
          DEF    L:44S4      L S M        347A
          DEF    L:44S3      L S M        3470
          DEF    L:44S2      L S M        3474
          DEF    L:44S1      L S M        3478
          DEF    L:44A4      L S M        3486
          DEF    L:44A3      L S M        347C
          DEF    L:44A2      L S M        3480
          DEF    L:43R3      L S M        341D
          DEF    L:43R2      L S M        3419
          DEF    L:43R1      L S M        3421
          DEF    L:43R4      L S M        3423
          DEF    L:43L4      L S M        3465
          DEF    ::DBLE      L S M        3416
          DEF    L:43L3      L S M        345F
          DEF    L:43L1      L S M        3463
          DEF    L:43C       L S M        3469
          DEF    :DBLE       L S M        3416
          DEF    L:34R4      L S M        33C3
          DEF    L:34R3      L S M        33BD
          DEF    L:34R1      L S M        33C1
          DEF    ::SNGL      L S M        33B6
          DEF    L:34L4      L S M        3407
          DEF    L:34L3      L S M        3401
          DEF    L:34L2      L S M        33FD
          DEF    L:34L1      L S M        3405
          DEF    L:34C       L S M        340D
          DEF    :SNGL       L S M        33B6
       P  REF    M:DEFINE    L S M        0000
       P  REF    M:OPEN      L S M        0000
          DEF    QREW        L S M        3342
          DEF    QFSKIP      L S M        3329
          DEF    QWEOF       L S M        3354
          DEF    QSKIP       L S M        32FA
          DEF    L:43L2      L S M        345B
```

```
        DEF     L:4H        L S M        35FE
        DEF     L:44R3      L S M        35E0
        DEF     L:44A1      L S M        3484
        DEF     L:44L3      L S M        350D
        DEF     L:44L2      L S M        3511
        DEF     L:44M3      L S M        352B
        DEF     L:34R2      L S M        33B9
        DEF     L:FMTØ      L S M        2F38
        DEF     L:FMTI      L S M        2DA4
        DEF     L:44M4      L S M        3531
        DEF     F:STCHR     L S M        2D6E
        DEF     F:CPWRT     L S M        2C9D
        DEF     F:FMTERR    L S M        2D30
        DEF     F:FIØRET    L S M        2C13
        DEF     L:FIØ       L S M        2B2B
P       REF     M:SR        L S M        0000
        DEF     L:XMT1      L S M        2AC0
        DEF     L:XMT       L S M        2AB9
        DEF     L:CLEAR     L S M        3606
        DEF     L:WSEQ      L S M        2A11
        DEF     L:IØSQER    L S M        2A36
P       REF     L:ERRØR     L S M        0000
P       REF     M:PUSH      L S M        0000
        DEF     L:FIØØP     L S M        2B28
        DEF     L:WSEQFN    L S M        2A9F
        DEF     L:WSEQI     L S M        2A13
        DEF     L:SEQRST    L S M        2978
        DEF     L:RSEQFN    L S M        2A9D
        DEF     L:FIØIN     L S M        2B25
        DEF     L:RSEQ      L S M        2A00
        DEF     L:SEQST     L S M        2947
        DEF     L:33M4      L S M        28C1
        DEF     L:33M2      L S M        28B7
        DEF     L:33M1      L S M        28BF
        DEF     L:33L3      L S M        28A7
        DEF     L:33L2      L S M        28AB
        DEF     L:33S3      L S M        280E
        DEF     L:33S2      L S M        2815
        DEF     L:33S1      L S M        281C
        DEF     L:33A2      L S M        2817
        DEF     L:33A1      L S M        281E
        DEF     L:33S4      L S M        280C
        DEF     L:33A4      L S M        280A
P       REF     M:PØP       I            0000
P       REF     L:88X       I            0000
P       REF     L:33R2      I            0000
        DEF     L:33A3      L S M        2810
        DEF     L:33M3      L S M        28BB
        DEF     L:33L1      L S M        28AF
P       REF     L:33R3      I            0000
P       REF     ::FLØAT     I            0000
        DEF     ØREAD       L S M        32AC
        DEF     ØCLØSE      L S M        3283
        DEF     ØWRITE      L S M        325C
        DEF     ØØPEN       L S M        31D8
        DEF     L:88W       L S M        2AB9
        DEF     L:88C2      L S M        2C86
        DEF     L:88S2      L S M        293E
        DEF     L:88C       L S M        2C86
        DEF     L:88S       L S M        2935
        DEF     DEFINE      L S M        3388
P       REF     M:PSHC      I            0000
```

```
            DEF    CINPUT      I             2648

SEGMENT IDENT CODE  ORG   LWA   LEN   TRA   REV
        0002  0000  2648  38DB  1294  0000  0000

            DEF    L:88W       L S M         3873
            DEF    L:XMT       L S M         3873
            DEF    L:44T4      L S M         3849
            DEF    L:44T2      L S M         383F
            DEF    L:44T1      L S M         3847
            DEF    L:43R3      L S M         37EE
            DEF    L:43R2      L S M         37EA
            DEF    L:43R1      L S M         37F2
            DEF    L:43R4      L S M         37F4
            DEF    ::DBLE      L S M         37E7
            DEF    L:43L3      L S M         3830
            DEF    L:43L1      L S M         3834
            DEF    L:43C       L S M         383A
            DEF    :DBLE       L S M         37E7
            DEF    L:34R4      L S M         3794
            DEF    L:34R3      L S M         378E
            DEF    L:34R1      L S M         3792
            DEF    ::SNGL      L S M         3787
            DEF    L:34L4      L S M         37D8
            DEF    L:34L3      L S M         37D2
            DEF    L:34L2      L S M         37CE
            DEF    L:34L1      L S M         37D6
            DEF    :SNGL       L S M         3787
            DEF    QREW        L S M         373B
            DEF    QWEOF       L S M         374D
            DEF    QWRITE      L S M         3655
            DEF    L:43L2      L S M         382C
            DEF    L:44T3      L S M         3843
            DEF    L:34R2      L S M         378A
            DEF    L:FMTO      L S M         3331
            DEF    L:FMTI      L S M         319D
            DEF    L:XMT1      L S M         387A
            DEF    F:STCHR     L S M         3167
            DEF    F:CPWRT     L S M         3096
            DEF    L:88C       L S M         307F
            DEF    F:FMTERR    L S M         3129
            DEF    F:FIØRET    L S M         300C
            DEF    L:FIØIN     L S M         2F1E
            DEF    L:FIØ       L S M         2F24
            DEF    L:RSEQFN    L S M         2F02
            DEF    L:CLEAR     L S M         3868
            DEF    L:WSEQ      L S M         2E76
            DEF    L:IOSQER    L S M         2E9B
            DEF    L:SEQRST    L S M         2DFE
            DEF    L:FIØOP     L S M         2F21
            DEF    L:WSEQFN    L S M         2F04
            DEF    L:WSEQI     L S M         2E78
            DEF    L:SEQST     L S M         2DBD
            DEF    L:44R2      L S M         2D92
            DEF    L:44R1      L S M         2D96
            DEF    L:44M4      L S M         2CDF
            DEF    L:44M2      L S M         2CD5
            DEF    L:44L4      L S M         2CC5
            DEF    L:44L3      L S M         2CBB
```

```
     DEF    L:44L2     L  S  M     2CBF
     DEF    L:44S4     L  S  M     2C51
     DEF    L:44S3     L  S  M     2C47
     DEF    L:44S2     L  S  M     2C4B
     DEF    L:44S1     L  S  M     2C4F
     DEF    L:44A4     L  S  M     2C5D
     DEF    L:44A3     L  S  M     2C53
     DEF    L:44A2     L  S  M     2C57
     DEF    L:34C      L  S  M     37DE
     DEF    L:4N       L  S  M     2DAC
     DEF    X:3NORM    L  S  M     2D52
P    REF    L:ERROR    L  S  M     0000
     DEF    L:44M3     L  S  M     2CD9
     DEF    L:44M1     L  S  M     2CD0
     DEF    L:44A1     L  S  M     2C5B
     DEF    L:44R3     L  S  M     2D8E
     DEF    L:44L1     L  S  M     2CC3
     DEF    L:33M4     L  S  M     2B21
     DEF    L:33L2     L  S  M     2B0B
     DEF    L:33D4     L  S  M     2A93
     DEF    L:33D2     L  S  M     2A89
     DEF    L:33D1     L  S  M     2A91
     DEF    L:33T3     L  S  M     2A76
     DEF    L:33T1     L  S  M     2A7A
     DEF    L:33S3     L  S  M     29D9
     DEF    L:33S1     L  S  M     29E7
     DEF    L:33A2     L  S  M     29E2
     DEF    L:33A1     L  S  M     29F9
     DEF    L:33S4     L  S  M     29D7
     DEF    L:33A4     L  S  M     29D5
     DEF    :DSIN:     L  S  M     2BBA
     DEF    L:43L4     L  S  M     3836
     DEF    :RECNVRT   L  S  M     2C42
P    REF    M:PUSH     L  S  M     0000
P    REF    M:SR       L  S  M     0000
     DEF    :ABS       L  S  M     29AE
     DEF    L:3N       L  S  M     2DAC
     DEF    L:33S2     L  S  M     29E0
     DEF    L:33M1     L  S  M     2B1F
     DEF    L:33L3     L  S  M     2B07
     DEF    SIN        L  S  M     29B9
     DEF    L:33M2     L  S  M     2B17
     DEF    COS.       L  S  M     29BB
P    REF    L:32L3     I           0000
     DEF    L:33D3     L  S  M     2A8D
P    REF    L:32C      I           0000
     DEF    OUT60      I           2BB3
     DEF    L:33T2     L  S  M     2A72
     DEF    L:33A3     L  S  M     29DB
     DEF    L:33M3     L  S  M     2B1B
     DEF    L:33L1     L  S  M     2B0F
     DEF    ::ABS      L  S  M     29AE
P    REF    L:33R3     I           0000
P    REF    M:POP      I           0000
     DEF    GCLOSE     L  S  M     367C
P    REF    L:88X      I           0000
     DEF    L:88C2     L  S  M     307F
     DEF    L:88S2     L  S  M     2DB4
     DEF    QSKIP      L  S  M     36F3
```

```
        P   REF    L:33R2      I             0000
        P   REF    ::FLØAT     I             0000
            DEF    ØREAD       L S M         36A5
            DEF    FIND        I             27DD
        P   REF    L:88Z       I             0000
            DEF    QFSKIP      L S M         3722
            DEF    QØPEN       L S M         35D1
        P   REF    M:PSHC      I             0000
            DEF    TINPUT      I             264B

SEGMENT IDENT NØDE  ØRG   LWA   LEN   TRA   SEV
         0003 0000  264B  34F9  0EAF  0000  0000

            DEF    L:88W       L S M         348E
            DEF    L:XMT       L S M         348E
            DEF    L:44R2      L S M         3469
            DEF    L:44R1      L S M         346D
            DEF    L:44M2      L S M         33AC
            DEF    L:44M1      L S M         33B4
            DEF    L:44L4      L S M         339C
            DEF    L:44L1      L S M         339A
            DEF    L:44T4      L S M         3373
            DEF    L:44T2      L S M         3369
            DEF    L:44T1      L S M         3371
            DEF    X:3NØRM     L S M         3429
            DEF    L:44S4      L S M         32FF
            DEF    L:44S3      L S M         32F5
            DEF    L:44S2      L S M         32F9
            DEF    L:44S1      L S M         32FD
            DEF    L:44A4      L S M         330B
            DEF    L:44A3      L S M         3301
            DEF    L:44A2      L S M         3305
            DEF    L:43R3      L S M         32A2
            DEF    L:43R2      L S M         329E
            DEF    L:43R1      L S M         32A6
            DEF    L:43R4      L S M         32A8
            DEF    L:43L4      L S M         32EA
            DEF    ::DBLE      L S M         329B
            DEF    L:43L3      L S M         32E4
            DEF    L:43L1      L S M         32E8
            DEF    L:43C       L S M         32EE
            DEF    :DBLE       L S M         329B
            DEF    L:34R4      L S M         3248
            DEF    L:34R3      L S M         3242
            DEF    L:34R1      L S M         3246
            DEF    ::SNGL      L S M         323B
            DEF    L:34L4      L S M         328C
            DEF    L:34L3      L S M         3286
            DEF    L:34L2      L S M         3282
            DEF    L:34L1      L S M         328A
            DEF    L:34C       L S M         3292
            DEF    :SNGL       L S M         323B
            DEF    L:43L2      L S M         32E0
            DEF    L:44T3      L S M         336D
            DEF    L:44R3      L S M         3465
            DEF    L:44A1      L S M         3309
            DEF    L:44L3      L S M         3392
            DEF    L:44L2      L S M         3396
            DEF    L:44M3      L S M         33B0
            DEF    L:34R2      L S M         323E
            DEF    L:FMT0      L S M         2FC7
            DEF    L:FMT1      L S M         [illegible]
         DEF   L:43M4     L S M        3386
         DEF   L:XCT1     L S M        3495
         DEF   F:STCHR    L S M        2DFD
         DEF   F:CPWRT    L S M        2D2C
         DEF   L:88C      L S M        2D15
         DEF   F:FMTERR   L S M        2D8F
         [illegible]   F:FIOSET   L S M        2CA[illegible]
         DEF   L:FIOIN    L S M        2BB4
         DEF   L:FIO      L S M        2BBA
         DEF   L:RSEQFN   L S M        2B98
         DEF   L:CLEAR    L S M        3483
         DEF   L:WSEQ     L S M        2B0C
         DEF   L:IOSQER   L S M        2B31
     P   REF   M:PUSH     L S M        0000
         DEF   L:SEQRST   L S M        2A84
         DEF   L:FIOOP    L S M        2BB7
         DEF   L:WSEQFN   L S M        2B9A
         DEF   L:WSEQI    L S M        2B0E
         DEF   L:SEQST    L S M        2A53
         DEF   L:4N       L S M        2A42
         DEF   L:33M4     L S M        29CE
         DEF   L:33M3     L S M        29C8
         DEF   L:33M1     L S M        29CC
         DEF   L:33D4     L S M        2940
         DEF   L:33D2     L S M        2936
         DEF   L:33T3     L S M        2923
         DEF   L:33T1     L S M        2927
         DEF   L:33S2     L S M        288D
         DEF   L:33S1     L S M        2894
         DEF   L:33S4     L S M        2884
         DEF   L:33A4     L S M        2882
     P   REF   L:ERROR    L S M        0000
     P   REF   M:SR       L S M        0000
     P   REF   M:POP      I            0000
         DEF   SQRT       L S M        2826
     P   REF   L:32R3     I            0000
         DEF   L:33A3     L S M        2888
         DEF   L:33M2     L S M        29C4
     P   REF   L:33R2     I            0000
         DEF   L:33L2     L S M        29B8
         DEF   L:3N       L S M        2A42
         DEF   L:33D3     L S M        293A
         DEF   L:33S3     L S M        2886
         DEF   L:33D1     L S M        293E
         DEF   L:33A2     L S M        288F
     P   REF   L:88X      I            0000
         DEF   L:88C2     L S M        2D15
         DEF   L:88S2     L S M        2A4A
         DEF   L:33T2     L S M        291F
         DEF   L:33A1     L S M        2896
         DEF   L:33L3     L S M        29B4
     P   REF   L:33R3     I            0000
         DEF   L:33L1     L S M        29BC
     P   REF   L:33R1     I            0000
     P   REF   ::FLOAT    I            0000
     P   REF   M:PSHC     I            0000
         DEF   DCAL       I            264B

SEGMENT IDENT NODE  ORG   LWA   LEN   TRA   SEV
         0004  0000  264B  3AD4  148A  0000  0000

         DEF   M:PUSHK    L S M        3AA4
         DEF   L:23E3     L S M        3A4D
```

```
       DEF    L:23E1     L S M      3A49
       DEF    :AINT      L S M      3A21
       DEF    L:AINT     L S M      3A21
       DEF    C:HARC     L S M      33CF
P      REF    M:WAIT     L S M      0000
P      REF    M:DOW      L S M      0000
       DEF    OTBM1      L S M      37C3
       DEF    BYTP       L S M      37C1
       DEF    OTB        L S M      37C4
P      REF    M:IOEX     L S M      0000
       DEF    L:23E2     L S M      3A4B
       DEF    ::AINT     L S M      3A21
P      REF    M:PUSH     L S M      0000
       DEF    VCTOI      L S M      35EE
       DEF    OUTNUM     L S M      3631
       DEF    MODRER     L S M      39F2
       DEF    L:4N       L S M      2EAF
       DEF    L:33M4     L S M      2E38
       DEF    L:33M3     L S M      2E35
       DEF    L:33D4     L S M      2DAD
       DEF    L:33D3     L S M      2DA7
       DEF    L:33T3     L S M      2D90
       DEF    L:33T2     L S M      2D8C
       DEF    L:33S1     L S M      2D01
       DEF    L:33S4     L S M      2CF1
       DEF    L:33A4     L S M      2CEF
P      REF    M:SR       L S M      0000
       DEF    L:23R4     L S M      2C9F
       DEF    L:23L4     L S M      2CB3
       DEF    L:23R3     L S M      2C99
       DEF    ::INT      L S M      2C92
       DEF    L:23R2     L S M      2C95
       DEF    L:23R1     L S M      2C9D
       DEF    L:23L3     L S M      2CAD
       DEF    L:23L2     L S M      2CA9
       DEF    L:23L1     L S M      2CB1
       DEF    L:23C      L S M      2CB7
       DEF    :IFIX      L S M      2C92
       DEF    :INT       L S M      2C92
       DEF    IN         L S M      374B
       DEF    SCRIBE     L S M      383F
       DEF    OUT        L S M      378B
       DEF    MODE       L S M      35BF
       DEF    LEGEND     L S M      32CE
       DEF    PASTOR     L S M      3883
       DEF    VCTOR      L S M      35D8
       DEF    SCALE      L S M      3511
       DEF    WORDXY     L S M      3A02
       DEF    IWRDXY     L S M      39E4
       DEF    POINTS     L S M      332A
P      REF    M:POP      I          0000
       DEF    LINES      L S M      33B9
       DEF    L:33M2     L S M      2E31
       DEF    L:33D2     L S M      2DA3
       DEF    L:33A1     L S M      2D03
       DEF    L:33T1     L S M      2D94
       DEF    L:33L1     L S M      2E29
       DEF    PAGE       L S M      3447
       DEF    BELL       I          2C4B
```

```
              DEF    MODLIN      I               2B69
              DEF    LABEL       L S M           3195
              DEF    GRID        L S M           3016
              DEF    SUBJEC      L S M           2F93
          P   REF    L:88Z       I               0000
              DEF    NUMBER      L S M           32FC
              DEF    WORDS       I               2BE2
              DEF    ØBJECT      L S M           2F20
              DEF    MØCSET      L S M           347E
              DEF    INITAL      L S M           2EB7
              DEF    L:33S3      L S M           2CF3
              DEF    L:33A2      L S M           2CFC
              DEF    L:33A3      L S M           2CF5
              DEF    L:33L3      L S M           2E21
              DEF    L:3N        L S M           2EAF
              DEF    L:33S2      L S M           2CFA
              DEF    L:33M1      L S M           2E39
              DEF    ::IFIX      L S M           2C92
              DEF    L:33D1      L S M           2DAB
              DEF    L:33L2      L S M           2E25
          P   REF    L:33R1      I               0000
          P   REF    L:33R2      I               0000
          P   REF    L:33R3      I               0000
          P   REF    ::FLØAT     I               0000
          P   REF    M:PSHC      I               0000
              DEF    GRAPH       I               264B

SEGMENT IDENT NODE  ØRG   LWA   LEN   TRA   SEV
        0005  0000  264B  2A88  043E  0000  0000

              DEF    L:4N        L S M           2A81
              DEF    L:33M4      L S M           2A0D
              DEF    L:33D4      L S M           297F
              DEF    L:33T1      L S M           2966
              DEF    L:33S2      L S M           28CC
              DEF    L:33S4      L S M           28C3
              DEF    L:33A4      L S M           28C1
              DEF    L:23R4      L S M           2871
              DEF    L:23L4      L S M           2885
              DEF    L:23R3      L S M           286B
              DEF    ::INT       L S M           2864
              DEF    L:23R2      L S M           2867
              DEF    L:23R1      L S M           286F
              DEF    L:23L3      L S M           287F
              DEF    L:23L2      L S M           287B
              DEF    L:23L1      L S M           2883
              DEF    L:23C       L S M           2889
              DEF    :IFIX       L S M           2864
              DEF    :INT        L S M           2864
              DEF    L:3N        L S M           2A81
          P   REF    M:SR        L S M           0000
              DEF    :ABS        L S M           2857
              DEF    ::IFIX      L S M           2864
              DEF    L:33T2      L S M           295E
              DEF    L:33D2      L S M           2975
              DEF    FØRCE       I               2787
          P   REF    M:PØP       I               0000
              DEF    L:33S1      L S M           28D3
              DEF    L:33A1      L S M           28D5
          DEF    L:33M2     L S M       2A03
          DEF    L:33M3     L S M       2A07
          DEF    L:33A3     L S M       28C7
          DEF    L:33T3     L S M       2962
          DEF    ::ABS      L S M       2857
       P  REF    L:33R2     I           0000
          DEF    L:33D1     L S M       297D
          DEF    L:33S3     L S M       28C5
          DEF    L:33L2     L S M       29F7
       P  REF    L:33R1     I           0000
          DEF    L:33D3     L S M       2979
          DEF    L:33M1     L S M       2A0B
       P  REF    ::FLOAT    I           0000
          DEF    L:33A2     L S M       28CE
          DEF    L:33L3     L S M       29F3
       P  REF    L:33R3     I           0000
          DEF    L:33L1     L S M       29FB
       P  REF    M:PSHC     I           0000
          DEF    ENG        I           264B

SEGMENT IDENT MODE  ORG   LWA   LEN   TRA   SEV
        0006  0000  264B  397B  1331  0000  0000

          DEF    L:43R3     L S M       392B
          DEF    L:43R2     L S M       3927
          DEF    L:43R1     L S M       392F
          DEF    L:43R4     L S M       3931
          DEF    ::DBLE     L S M       3924
          DEF    L:43L3     L S M       396D
          DEF    L:43L2     L S M       3969
          DEF    L:43L1     L S M       3971
          DEF    L:43C      L S M       3977
          DEF    :DBLE      L S M       3924
          DEF    L:34R4     L S M       38D1
          DEF    L:34R3     L S M       38CB
          DEF    L:34R2     L S M       38C7
          DEF    L:34R1     L S M       38CF
          DEF    ::SNGL     L S M       38C4
          DEF    L:34L4     L S M       3915
          DEF    L:34L3     L S M       390F
          DEF    L:34L2     L S M       390B
          DEF    L:34L1     L S M       3913
          DEF    :SNGL      L S M       38C4
          DEF    QREW       L S M       3878
          DEF    QSKIP      L S M       3830
          DEF    QREAD      L S M       37E2
          DEF    L:22E1     L S M       36AD
          DEF    ::IA:S     L S M       369D
          DEF    L:44R2     L S M       3683
          DEF    L:44R1     L S M       3687
          DEF    L:44M4     L S M       35D0
          DEF    L:44M2     L S M       35C6
          DEF    L:44L4     L S M       35B6
          DEF    L:44L3     L S M       35AC
          DEF    L:44L2     L S M       35B0
          DEF    L:44S4     L S M       3542
          DEF    L:44S3     L S M       3538
          DEF    L:44S2     L S M       353C
          DEF    L:44S1     L S M       3540
          DEF    L:44A4     L S M       354E
          DEF    L:44A3     L S M       3544
        DEF   L:34C        L S M     391B
        DEF   L:4N         L S M     36D4
        DEF   X:3NORM      L S M     3643
        DEF   L:44M3       L S M     35CA
        DEF   L:44M1       L S M     35CE
        DEF   L:44A1       L S M     354C
        DEF   L:44P3       L S M     357F
        DEF   L:44L1       L S M     35B4
        DEF   L:33M4.      L S M     3412
        DEF   L:33D4       L S M     3384
        DEF   L:33T1       L S M     3368
        DEF   L:33S4       L S M     32C8
        DEF   L:33A4       L S M     32C6
  P     REF   L:ERRØR      L S M     0000
        DEF   :DSIN:       L S M     34A8
        DEF   L:43L4       L S M     3973
        DEF   :RECNVRT     L S M     3533
  P     REF   M:PUSH       L S M     0000
        DEF   L:23R4       L S M     31F4
        DEF   L:23L4       L S M     3208
        DEF   L:23R3       L S M     31EE
        DEF   ::INT        L S M     31E7
        DEF   L:23R2       L S M     31EA
        DEF   L:23R1       L S M     31F2
        DEF   L:23L3       L S M     3202
        DEF   L:23L2       L S M     31FE
        DEF   L:23L1       L S M     3206
        DEF   L:23C        L S M     320C
        DEF   :IFIX        L S M     31E7
        DEF   :INT         L S M     31E7
  P     REF   M:SR         L S M     0000
        DEF   :ABS         L S M     31DA
        DEF   QCLØSE       L S M     37B9
        DEF   QWEØF        L S M     388A
        DEF   QWRITE       L S M     3792
        DEF   QFSKIP       L S M     385F
        DEF   QØPEN        L S M     370E
        DEF   SIN          L S M     3245
  P     REF   L:32R3       I         0000
        DEF   ::IABS       L S M     369D
        DEF   L:33S3       L S M     32CA
        DEF   L:33S2       L S M     32D1
        DEF   FØRT         I         28F2
  P     REF   L:32L1       I         0000
        DEF   ::ABS        L S M     31DA
        DEF   L:22E2       L S M     36A5
        DEF   L:3N         L S M     36D4
        DEF   CØS          L S M     3247
  P     REF   M:PØP        I         0000
        DEF   SQRT         L S M     326A
        DEF   L:33S1       L S M     32D8
        DEF   L:33A1       L S M     32DA
        DEF   L:33D1       L S M     3382
  P     REF   L:33R1       I         0000
        DEF   L:33M1       L S M     3410
        DEF   L:33D2       L S M     337A
        DEF   L:33T2       L S M     3363
        DEF   L:33T3       L S M     3367
        DEF   L:33A2       L S M     32D3
        DEF   ::IFIX       L S M     31E7
        DEF   L:33L3       L S M     33F8
        DEF   PSDSAV       I         3143
  P     REF   L:33R2       I         0000
```

```
             DEF     L:33A3      L S M       32CC
             DEF     L:33M2      L S M       3408
             DEF     L:33L2      L S M       33FC
             DEF     L:33M3      L S M       340C
             DEF     L:33D3      L S M       337E
             DEF     L:33L1      L S M       3400
             DEF     RFØRT       I           294E
          P  REF     L:33R3      I           0000
          P  REF     ::FLØAT     I           0000
             DEF     L:22E3      L S M       36A9
             DEF     WINDØW      I           28B2
          P  REF     M:PSHC      I           0000
             DEF     FFTPSD      I           264B

SEGMENT IDENT NØDE  ØRG   LWA   LEN   TRA   SEV
        0007  0000  264B  3ABA  1470  0000  0000

             DEF     L:44R2      L S M       3A96
             DEF     L:44R1      L S M       3A9A
             DEF     L:44M2      L S M       39D9
             DEF     L:44M1      L S M       39E1
             DEF     L:44L4      L S M       39C9
             DEF     L:44L1      L S M       39C7
             DEF     L:44T4      L S M       39A0
             DEF     L:44T2      L S M       3996
             DEF     L:44T1      L S M       399E
             DEF     X:3NØRM     L S M       3A56
             DEF     L:44S4      L S M       392C
             DEF     L:44S3      L S M       3922
             DEF     L:44S2      L S M       3926
             DEF     L:44S1      L S M       392A
             DEF     L:44A4      L S M       3938
             DEF     L:44A3      L S M       392E
             DEF     L:44A2      L S M       3932
             DEF     L:43R3      L S M       38CF
             DEF     L:43R2      L S M       38CB
             DEF     L:43R1      L S M       38D3
             DEF     L:43R4      L S M       38D5
             DEF     L:43L4      L S M       3917
             DEF     ::DBLE      L S M       38C8
             DEF     L:43L3      L S M       3911
             DEF     L:43L1      L S M       3915
             DEF     L:43C       L S M       391B
             DEF     :DBLE       L S M       38C8
             DEF     L:34R4      L S M       3875
             DEF     L:34R3      L S M       386F
             DEF     L:34R1      L S M       3873
             DEF     ::SNGL      L S M       3868
             DEF     L:34L4      L S M       38B9
             DEF     L:34L3      L S M       38B3
             DEF     L:34L2      L S M       38AF
             DEF     L:34L1      L S M       38B7
             DEF     L:34C       L S M       38BF
             DEF     :SNGL       L S M       3868
             DEF     L:43L2      L S M       390D
             DEF     L:44T3      L S M       399A
             DEF     L:44R3      L S M       3A92
             DEF     L:44A1      L S M       3936
             DEF     L:44L3      L S M       39BF
```

C2

```
    DEF   L:44L2    L S M    39C3
    DEF   L:44M3    L S M    39DD
    DEF   L:34R2    L S M    336D
    DEF   L:FMTØ    L S M    35F4
    DEF   L:FMTI    L S M    3460
    DEF   L:44M4    L S M    39E3
    DEF   F:STCHR   L S M    342A
    DEF   F:CPWRT   L S M    3359
    DEF   L:88C     L S M    3342
    DEF   F:FMTERR  L S M    33EC
    DEF   F:FIØRET  L S M    32CF
    DEF   L:FIØIN   L S M    31E1
    DEF   L:FIØ     L S M    31E7
    DEF   L:XMT1    L S M    317C
    DEF   L:XMT     L S M    3175
    DEF   L:RSEQFN  L S M    3159
    DEF   L:CLEAR   L S M    3AB0
    DEF   L:WSEQ    L S M    30CD
    DEF   L:IØSQER  L S M    30F2
P   REF   L:ERRØR   L S M    0000
P   REF   M:PUSH    L S M    0000
    DEF   L:SEQRST  L S M    3045
    DEF   L:FIØØP   L S M    31E4
    DEF   L:WSEQFN  L S M    3158
    DEF   L:WSEQI   L S M    30CF
    DEF   L:SEQST   L S M    3014
    DEF   L:4N      L S M    3003
    DEF   L:33M4    L S M    2F8F
    DEF   L:33M3    L S M    2F89
    DEF   L:33D4    L S M    2F01
    DEF   L:33D3    L S M    2EFB
    DEF   L:33S3    L S M    2E5E
    DEF   L:33S1    L S M    2E6C
    DEF   L:33A1    L S M    2E6E
    DEF   L:33S4    L S M    2E5C
    DEF   L:33A4    L S M    2E5A
P   REF   M:SR      L S M    0000
    DEF   L:23R4    L S M    2E0A
    DEF   L:23L4    L S M    2E1E
    DEF   L:23R3    L S M    2E04
    DEF   ::INT     L S M    2DFD
    DEF   L:23R2    L S M    2E00
    DEF   L:23R1    L S M    2E08
    DEF   L:23L3    L S M    2E18
    DEF   L:23L2    L S M    2E14
    DEF   L:23L1    L S M    2E1C
    DEF   L:23C     L S M    2E22
    DEF   :IFIX     L S M    2DFD
    DEF   :INT      L S M    2DFD
P   REF   M:PØP     I        0000
P   REF   L:88X     I        0000
    DEF   L:33M2    L S M    2F85
    DEF   L:33A2    L S M    2E67
    DEF   L:33D2    L S M    2EF7
    DEF   L:33L1    L S M    2F7D
    DEF   L:88W     L S M    3175
    DEF   L:88C2    L S M    3342
    DEF   L:88S2    L S M    300B
    DEF   L:33A3    L S M    2E60
```

```
       DEF   L:33L3     L S M     2F75
       DEF   L:3N       L S M     3003
       DEF   L:33S2     L S M     2E65
       DEF   L:33H1     L S M     2F8D
    P  REF   ::FLOAT    I         0000
       DEF   ::IFIX     L S M     2DFD
    P  REF   L:33R3     I         0000
       DEF   L:33D1     L S M     2EFF
       DEF   L:33L2     L S M     2F79
    P  REF   M:PSHC     I         0000
       DEF   PRINT      I         264B

ERRSEV= 0000

END MAP
```

APPENDIX G

NOVA SOURCE LISTING AND LOAD MAP

```
C***********************************************************************
C
C
C       PROGRAM:     EMGAN
C
C       AUTHOR:      WILLIAM M. HURSTA
C
C       PURPOSE:     TO PROVIDE  FIRST ORDER REDUCTION OF EMG AND FORCE DATA.
C
C***********************************************************************
C
C
      EXTERNAL OV1,OV2,OV3,OV4,OV5,OV6,OV7
      INTEGER DTYPE,FILNUM,FILTSW,PLOTSW
      INTEGER FCALSW ,ECALSW,WHOA,PRNTSW,OVL
      DIMENSION ITFILE(4)
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (DTYPE,IVARSW(1));(FILNUM,IVARSW(2))
      EQUIVALENCE (FILTSW,IVARSW(3)),(NSECS,IVARSW(4))
      EQUIVALENCE (PLOTSW,IVARSW(5)),(PRNTSW,IVARSW(6))
      EQUIVALENCE (ISPAN,IVARSW(7)),(IFGAIN,IVARSW(8))
      MTO=2
      OVL=0
      FCALSW=0
      ECALSW=0
      WHOA=0
      NSLICE=0
C
C      INITIALIZE MAG TAPE UNITS AND OPEN PSD OUTPUT TAPE FILE
C
      CALL INIT ("MT0",0,IER)
      CALL INIT ("MT1",0,IER)
      CALL MTOPD (MTO,"MT1:0",0,IER)
C
C      OPEN OVERLAY FILE
C
      CALL OVOPN (OVL,"EMGAN.OL",IER)
C
C     READ IN DATA CARDS
C
      CALL OVLOD (OVL,OV1,0,IER)
      CALL CINPUT (NSLICE,I,START)
      DO 200 I=1,NSLICE
C
C     GO GET DATA CARD INFO FOR PRESENT DATA SLICE
C
      CALL OVLOD (OVL,OV1,0,IER)
      CALL CINPUT (NSLICE,I,START)
      SPAN=FLOAT(ISPAN)
      IF(DTYPE.EQ.4) FDGAIN=FLOAT(IFGAIN)
C
C      CHECK TO SEE IF CALIBRATION DATA ACQUIRED BEFORE PROCESSING FIRST DAT
C      SLICE
C
      IF(DTYPE.EQ.1) FCALSW=1
      IF(DTYPE.EQ.2) ECALSW=1
      IF(DTYPE.EQ.3.AND.ECALSW .NE.1) WHOA=1
      IF(DTYPE.EQ.4.AND.FCALSW.NE.1) WHOA=1
      IF(WHOA.EQ.1) GO TO 90
C
C      GO GET THE DATA FROM TAPE
C
```

```
      CALL OVLOD (OVL,OV2,0,IER)
      CALL TINPUT(DTYPE,ITFILE,FILNUM,START,NSECS,I)
C
C       DECIDE WHERE TO GO FOR EACH DATA TYPE.
C
      GO TO (20,10,30,70),DTYPE
C
C       FILTER 60 HZ  FOR EMG CAL?
C
   10 IF(FILTSW.EQ.-1) GO TO 20
      CALL OUT60 (DTYPE,NSECS,FILTSW)
C
C       CALCULATE CALIBRATION VARIABLES OF EMG OR FORCE DATA.
C
   20 CALL OVLOD (OVL,OV3,0,IER)
      CALL DCAL (DTYPE,NSECS,IFGAIN)
C
C       PLOT EMG OR FORCE CAL DATA?
C
      IF(PLOTSW.EQ.1) GO TO 25
      CALL OVLOD (OVL,OV4,0,IER)
      CALL GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
C
C       PRINT OUT HEADER?
C
   25 IF(PRNTSW.EQ.1) GO TO 200
      GO TO 90
C
C       FILTER 60 HZ  FOR EMG DATA SLICE?
C
   30 IF(FILTSW.EQ.-1) GO TO 40
      CALL OUT60 (DTYPE,NSECS,FILTSW)
C
C       SCALE EMG DATA, CALCULATE INTEGRATED VALUE AND MEAN SQUARE VALUE
C
   40 CALL OVLOD (OVL,OV5,0,IER)
      CALL EMG (NSECS,VOLTSEC)
C
C       PLOT EMG DATA ?
C
      IF(PLOTSW.EQ.1) GO TO 50
      CALL OVLOD (OVL,OV4,0,IER)
      CALL GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
C
C       TAKE THE FOURIER TRANSFORM OF THE DATA AND FIND PSD UP TO 400 HZ.
C
   50 CALL OVLOD (OVL,OV6,0,IER)
      CALL FFTPSD (NSECS,SPAN,PSDSUM,STDERR)
      DTYPE=5
C
C       PLOT THE PSD ?
C
      IF(PLOTSW.EQ.1) GO TO 60
      CALL OVLOD (OVL,OV4,0,IER)
      CALL GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
C
C       PRINT OUT RESULTS ON LINE PRINTER?
C
   60 IF(PRNTSW.EQ.1) GO TO 200
      GO TO 90
C
C       SCALE FORCE DATA FOR PLOTTING AND AVERAGE IT FOR PRINTING.
C
```

```
   70 CALL OVLOD (OVL,OV5,0,IER)
      CALL FORCE (DTYPE,NSECS,SPAN,FDGAIN)
C
C      PLOT FORCE DATA ?
C
      IF(PLOTSW.EQ.1) GO TO 80
      CALL OVLOD (OVL,OV4,0,IER)
      CALL GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
C
C      PRINT OUT FORCE DATA ON LINE PRINTER?
C
   80 IF(PRNTSW.EQ.1) GO TO 200
   90 CALL OVLOD (OVL,OV7,0,IER)
      CALL POUT (DTYPE,START,NSECS,SPAN,VOLTSEC,PSDSUM,STDERR,WHOA)
  200 CONTINUE
      CALL RLSE ("MT0",IER)
      CALL RLSE ("MT1",IER)
      CALL CLOSE (OVL,IER)
      END
      OVERLAY OV1
      SUBROUTINE CINPUT (NSLICE,I,START)
C
C      SUBROUTINE CINPUT AQUIRES THE CARD DATA AND STORES IT ON A RAD FILE
C
C        ARGUMENTS:  NSLICE- NUMBER OF DATA SLICES
C
C                    I      - IDENTIFIES PRESENT DATA SLICE BEING PROCESSED
C
C                    START - BEGIN TIME OF PRESENT DATA SLICE
C
      DIMENSION ICARD(200,15)
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (ICARD,DATUMS(1))
      INTEGER CDR,DSK
      LPT=12
      CDR=9
      DSK=1
      IF(NSLICE) 120,20,60
C
C      INPUT NUMBER OF DATA SLICES AND READ DATA SLICE INFO FOR PRESENT RUN
C
   20 READ(CDR,30) NSLICE
   30 FORMAT(I5)
      WRITE(LPT,35) NSLICE
   35 FORMAT(1H1,' **** DATA CARDS ****',///,' NO. OF DATA SLICES=',I5,/
     @//,5X,'DATA SLICES',//)
      IF(NSLICE.GT.200) GO TO 100
      IF(NSLICE.LT.1) GO TO 120
      DO 50 J=1,NSLICE
      READ(CDR,40) (ICARD(J,K),K=1,15)
   40 FORMAT(1X,A2,14I3)
      WRITE(LPT,40) (ICARD(J,K),K=1,15)
   50 CONTINUE
      CALL FOPEN (DSK,"ICARD",6000)
      WRITE BINARY (DSK) ICARD
      CALL CLOSE (DSK,IER)
      GO TO 200
C
C      INITIALIZE VARIABLES AND SWITCHES FOR EACH DATA SLICE
C
   60 CALL FOPEN (DSK,"ICARD",6000)
      READ BINARY (DSK) ICARD
      CALL CLOSE (DSK,IER)
      IVARSW(1)=ICARD(I,2)
      IVARSW(2)=ICARD(I,1)
      START=3600.*FLOAT(ICARD(I,3))+60.*FLOAT(ICARD(I,4))+FLOAT(ICARD(I,
     @5))
      IVARSW(3)=ICARD(I,6)
      IVARSW(4)=ICARD(I,7)
      IVARSW(5)=ICARD(I,8)
      IVARSW(6)=ICARD(I,9)
      IVARSW(7)=ICARD(I,10)
      IVARSW(8)=ICARD(I,11)
      IHEAD(16)=ICARD(I,3)
      IHEAD(17)=ICARD(I,4)
      IHEAD(18)=ICARD(I,5)
      IHEAD(19)=ICARD(I,7)
      IF(IVARSW(1)-2) 90,90,200
   90 IHEAD(20)=ICARD(I,12)
      GO TO 200
C
```

```
C       ERROR MESSAGES
C
  100 WRITE(LPT,110) NSLICE
  110 FORMAT(///,' FATAL ERROR...READ NUMBER OF DATA SLICES TO BE',I6,'.
     & CANNOT HAVE MORE THAN 200.')
      STOP
  120 WRITE(LPT,130) NSLICE
  130 FORMAT(///,' FATAL ERROR...READ # OF DATA CARDS TO BE NEGATIVE.
     @# OF SLICES= ',I6)
      STOP
  200 IF(I.EQ.NSLICE) CALL DFILW ("ICARD",IER)
      RETURN

      END
      OVERLAY OV2
      SUBROUTINE TINPUT(DTYPE,ITFILE,FILNUM,START,NSECS,I)
C
C      SUBROUTINE TINPUT READS FORCE AND EMG DATA IN FROM TAPE.
C
C        CALLING ARGUMENTS:  DTYPE - DATA TYPE
C
C                            ITFILE- FILE CURRENTLY BEING ACCESSED ON TAPE
C
C                            FILNUM- FILE ON TAPE WHERE DESIRED DATA IS LOCA
C
C                            START - TIME IN TOTALED SECONDS OF BEGINNING OF
C                                    DESIRED DATA SLICE
C
C                            NSECS - NUMBER OF SECONDS OF DATA TO BE READ .
C
C                            I     - DATA SLICE NUMBER
C
      INTEGER FILNUM,DTYPE
      DIMENSION IARRAY(536),ITFILE(4),ITIME(3)
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (IARRAY,DATUMS(4230)),(ITIME,IARRAY(2))
      LPT=12
      MTI=3
      IREAD=1030K
      IGOOD=4105K
      IBREC=40000K
C
C      CHECK TO SEE IF DESIRED FILE IS CURRENTLY BEING ACCESSED.
C      IF NOT, EXCEPT ON FIRST DATA SLICE CLOSE PRESENT FILE BEFORE
C      OPENING NEW ONE.
C
      IF(I.EQ.1) GO TO 1
      IF(FILNUM.EQ.ITFILE(3)) GO TO 5
      CALL CLOSE (MTI,IER)
    1 ITFILE(1)="MT"
      ITFILE(2)="0:"
      ITFILE(3)=FILNUM
      ITFILE(4)="<0><0>"
C
C      POSITION TAPE AT THE BEGINNING OF THE DESIRED DATA FILE.
C
      CALL MTOPD (MTI,ITFILE,0,IER)
      IF(IER.NE.1) GO TO 300
C
C      POSITION TAPE AT THE BEGINNING OF THE DATA SLICE.
C
    5 CALL FIND (FILNUM,START)
C
C      DECIDE WHERE TO GO FOR EACH DATA TYPE.
C
      GO TO (10,50,50,10),DTYPE
C
C      ACQUIRE FORCE CAL OR FORCE DATA
C
   10 JJ=1
      NRECS=2*NSECS
      DO 30 J=1,NRECS
      CALL MTDIO (MTI,IREAD,IARRAY,ISTAT,IER,NWRDS)
      IF(IER.NE.1.OR.ISTAT.NE.IGOOD.OR.NWRDS.NE.535) GO TO 200
      DO 20 K=27,535,51
      DATUMS(JJ)=FLOAT(IARRAY(K))/8.
      JJ=JJ+1
   20 CONTINUE
   30 CONTINUE
      IF(DTYPE.EQ.1) GO TO 1000
      GO TO 150
C
C       ACQUIRE EMG CAL OR EMG DATA
C
   50 JJ=1
      NRECS=2*NSECS+1
      DO 80 J=1,NRECS
      CALL MTDIO (MTI,IREAD,IARRAY,ISTAT,IER,NWRDS)
      IF(IER.NE.1.OR.ISTAT.NE.IGOOD.OR.NWRDS.NE.535) GO TO 200
      DO 70 K=1,10
      L=25+K+50*(K-1)
      L50=L+50
      DO 60 KK=L,L50
      IF(KK.EQ.L+1) GO TO 60
      DATUMS(JJ)=FLOAT(IARRAY(KK))/8.
      JJ=JJ+1
   60 CONTINUE
   70 CONTINUE
   80 CONTINUE
      IF(DTYPE.EQ.2) GO TO 1000
  150 IBACK=IBREC+2*NSECS+2
      CALL MTDIO (MTI,IBACK,IARRAY,ISTAT,IER)
      GO TO 1000
C
C       ERROR MESSAGES
C
  200 WRITE(LPT,210) DTYPE,FILNUM,ISTAT,IER,NWRDS,ITIME
  210 FORMAT(///," FATAL ERROR...DURING INPUT OF DATA TYPE",I2,"IN FILE
     @",A2,/,10X,"TAPE STATUS(8)=",O18,/,10X,"FORTRAN ERROR=",I7,/,10X,"
     @# OF WORDS READ=",I6,/,10X,"DATA TIME=",4X,3I3)
      STOP
  300 WRITE(LPT,310) IER,FILNUM
  310 FORMAT(///," FATAL ERROR...FORTRAN ERROR CODE=",I3," DURING SEARCH
     @ FOR FILE ",A2)
      STOP
 1000 CONTINUE
      RETURN
      END
      SUBROUTINE FIND (FILNUM,START)
C
C       SUBROUTINE FIND POSITIONS THE TAPE AT THE BEGINNING OF A DATA SLICE.
C
C          CALLING ARGUMENTS:  FILNUM- FILE ON TAPE WHERE DATA SLICE IS LOCAT
C
C                              START - TIME IN TOTALED SECONDS OF THE BEGINNIN
C                                      DESIRED DATA SLICE.
C
C
      DIMENSION IARRAY(536)
      INTEGER FILNUM
      REAL NOW
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGO/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (IARRAY,DATUMS(4230))
      IPASS=0
      LPT=12
      MTI=3
      IEOF=-73273K
      IFFILE=30000K
      IBFILE=40000K
      IREAD=1030K
C
C       READ EACH RECORD AND COMPARE ITS TIME LABEL WITH THE START TIME.
C
   10 CALL MTDIO (MTI,IREAD,IARRAY,ISTAT,IER)
      IF(ISTAT.EQ.IEOF) GO TO 20
      NOW=3600.*ABS(FLOAT(IARRAY(2)))+60.*ABS(FLOAT(IARRAY(3)))+ABS(FLOA
     @T(IARRAY(4)))
      IF(NOW.NE.START) GO TO 10
C
C       ACQUIRE THE HEADER INFORMATION FOR THE PRESENT DATA SLICE.
C
      DO 15 J=1,15
      IHEAD(J)=IARRAY(J+5)
   15 CONTINUE
      IBREC=IBFILE+1
      CALL MTDIO (MTI,IBREC,IARRAY,ISTAT,IER)
      GO TO 100
C
C       DID NOT FIND THE START TIME BEFORE ENCOUNTERING AN EOF. IF ONLY ONE PA
C       HAS BEEN MADE THRU THE DATA, BACK UP AND TRY AGAIN. OTHERWISE,STOP]
C
   20 IF(IPASS.EQ.1) GO TO 40
      DO 30 J=1,2
   30 CALL MTDIO (MTI,IBFILE,IARRAY,ISTAT,IER)
      CALL MTDIO (MTI,IFFILE,IARRAY,ISTAT,IER)
      IPASS=1
      GO TO 10
   40 WRITE(LPT,50) FILNUM
   50 FORMAT(///,' CANNOT FIND START TIME IN FILE ',A2)
      STOP
  100 CONTINUE
      RETURN
      END
      SUBROUTINE OUT60 (DTYPE,NSECS,FILTSW)
C
C      SUBROUTINE OUT60 IS A DIGITAL NOTCH FILTER. DEPENDING ON THE DATA CARD
C        REQUEST, ONLY 60 HZ IS REMOVED OR ALL HARMONICS OF 60 HZ UP TO 360 H
C
C        CALLING ARGUMENTS:  DTYPE - TYPE OF DATA
C                            NSECS - NUMBER OF SECS OF DATA TO BE FILTERED
C
C                            FILTSW- SWITCH WHICH INDICATES WHETHER ONLY 60 H
C                                    IS TO BE REMOVED OR ALL HARMONICS OF 60
C
C
      INTEGER DTYPE, FILTSW
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (IRATE,IHEAD(11))
      LOOPS=1
      TWOPI=6.28318530717
      IF(FILTSW.EQ.1) LOOPS=5
      NPTS=1024*NSECS
      DO 30 J=1,LOOPS,2
      A=0.0
      B=0.0
      W=(60*J)*TWOPI/FLOAT(IRATE)
      DO 10 K=1,NPTS
      A=A+DATUMS(K)*COS(K*W)
      B=B+DATUMS(K)*SIN(K*W)
   10 CONTINUE
      A=A*2./FLOAT(NPTS)
      B=B*2./FLOAT(NPTS)
      DO 20 K=1,NPTS
      DATUMS(K)=DATUMS(K)-A*COS(K*W)-B*SIN(K*W)
   20 CONTINUE
   30 CONTINUE
      RETURN
      END
      OVERLAY OV3
      SUBROUTINE DCAL(DTYPE,NSECS,IFGAIN)
C
C      SUBROUTINE DCAL CALCULATES THE SCALE FACTORS FOR THE FORCE AND EMG DAT
C        THE CALIBRATION CONSTANT FOR INTEGRATED EMG AREA IS ALSO FOUND.  EM
C      THE CAL DATA IS THEN CONVERTED INTO UNITS OF POUNDS OR MICROVOLTS FOR
C      PLOTTING ON THE COMPUTEK TERMINAL.
C
C        CALLING ARGUMENTS:  DTYPE - TYPE OF DATA
C
C                            NSECS - LENGTH OF CAL DATA
C
C                            IFGAIN- GAIN APPLIED TO FORCE CAL SIGNAL
C
C
      DIMENSION HIGH(10),LOW(10),POUNDS(2600)
      INTEGER DTYPE,FLBCAL
      REAL LOWSUM
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (FLBCAL,IHEAD(15)),(IRATE,IHEAD(11))
      EQUIVALENCE (MAGTUD,IHEAD(13)),(POUNDS,DATUMS(1))
      LPT=12
      IF(DTYPE.EQ.2) GO TO 100
C
C       FIND FORCE CALIBRATION VARIABLES
C
      CAL(1,1)=FLOAT(FLBCAL)
      CAL(1,4)=FLOAT(IFGAIN)
C
C       LOOK FOR JUMP IN DATA TO INDICATE FORCE CAL.
C
      BEGIN=DATUMS(80)
      NPTS=20*NSECS-80
      DO 10 J=81,NPTS
      IF(DATUMS(J).GT.BEGIN+500.) GO TO 30
   10 CONTINUE
      WRITE (LPT,20)
   20 FORMAT(///,'COULD NOT FIND THE FORCE CAL.___CHECK DATA SLICE TIMES
     @')
      STOP
C
C      FOUND THE JUMP, NOW COMPUTE THE AVERAGE BEFORE AND AFTER THE JUMP.
C      CAL WEIGHT IN COUNTS = AVERAGE AFTER - AVERAGE BEFORE
C      FORCE SLOPE = COUNTS/POUND
C      ZERO FORCE  = AVERAGE COUNTS JUST BEFORE CAL JUMP
C
   30 HISUM=0.0
      LOWSUM=0.0
      DO 40 K=1,20
      HISUM=HISUM+DATUMS(J+K+60)
      LOWSUM=LOWSUM+DATUMS(J-K-60)
   40 CONTINUE
      FORSLP=(HISUM/20.-LOWSUM/20.)/FLOAT(FLBCAL)
      CAL(1,2)=FORSLP
      ZEROFOR=LOWSUM/20.
      CAL(1,3)=ZEROFOR
C
C      SCALE FORCE CALIBRATION DATA
C
      NPTS=20*NSECS
      DO 50 J=1,NPTS
      POUNDS(J)=(DATUMS(J)-ZEROFOR)/FORSLP
   50 CONTINUE
      GO TO 200
C
C        FIND EMG CALIBRATION VARIABLES
C        FIRST REMOVE ANY DC OFFSET IN THE CAL DATA
C
  100 SUM=0.0
      INDEX=1000*NSECS
      DO 110 J=1,INDEX
      SUM=SUM+DATUMS(J)
  110 CONTINUE
      OFFSET=SUM/FLOAT(INDEX)
      DO 120 J=1,INDEX
      DATUMS(J)=DATUMS(J)-OFFSET
  120 CONTINUE
C
C        CALCULATE STANDARD DEVIATION AND OBTAIN SCALE FACTOR. FOR ZERO MEAN,
C        STATIONARY SIGNAL STANDARD DEVIATION=RMS VALUE
C
      XSQUAR=0.0
      DO 140 J=1,INDEX
      XSQUAR=XSQUAR+DATUMS(J)**2
  140 CONTINUE
      EMGCAL=SQRT(XSQUAR/(INDEX-1))/FLOAT(MAGTUD)
      CAL(2,1)=FLOAT(MAGTUD)
      CAL(2,2)=EMGCAL
C
C        SCALE CAL DATA FOR PLOTTING.
C
      DO 180 J=1,INDEX
      DATUMS(J)=DATUMS(J)/EMGCAL
  180 CONTINUE
  200 RETURN
      END
      OVERLAY OV4
      SUBROUTINE GRAPH (DTYPE,START,NSECS,SPAN,STDERR)
C
C        SUBROUTINE GRAPH PLOTS ON THE COMPUTEK TERMINAL THE DATA SLICE BEING
C          PROCESSED ALONG WITH VARIOUS HEADER INFORMATION.
C
C          CALLING ARGUMENTS:  DTYPE - TYPE OF DATA
C
C                              START - BEGIN TIME OF PRESENT DATA SLICE
C
C                              NSECS - LENGTH OF DATA SLICE IN SECONDS
C
C                              SPAN  - AVERAGING INTERVAL FOR PSD
C
C                              STDERR- NORMALIZED STANDARD ERROR FOR PSD ESTIMA
C
      DIMENSION DATE(3),POUNDS(2600),PSDN(410)
      DIMENSION FREQ(410),CALDATA(2000)
      INTEGER DTYPE
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (POUNDS,DATUMS(1)),(PSDN,DATUMS(4000))
      EQUIVALENCE (FREQ,DATUMS(3500)),(FLBCAL,CAL(1,1))
      EQUIVALENCE (CALDATA,DATUMS(1)),(EMGMAX,DATUMS(4499))
      EQUIVALENCE (FORMAX,DATUMS(4500))
      RETURN
      END
      OVERLAY OV5
      SUBROUTINE EMG (NSECS,VOLTSEC)
C
C       SUBROUTINE EMG SCALES EMG DATA FOR PLOTTING ON THE COMPUTER AND FINDS
C         THE INTEGRATED EMG VALUE FOR THE DATA SLICE.
C
C         CALLING ARGUMENTS:  NSECS - LENGTH OF DATA SLICE
C
C                             VOLTSEC- INTEGRATED EMG VALUE IN MICROVOLT*SEC.
C
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (IRATE,IHEAD(11)),(EMGCAL,CAL(2,2))
      EQUIVALENCE (EMGMAX,DATUMS(4499)),(EMGVAR,DATUMS(4500))
C
C         CALCULATE OFFSET, SUBTRACT FROM DATA AND APPLY SCALE FACTOR
C
      SUM=0.0
      NPTS=1024*NSECS
      DO 10 J=1,NPTS
      SUM=SUM+DATUMS(J)
   10 CONTINUE
      OFFSET=SUM/FLOAT(NPTS)
      CAL(2,4)=OFFSET
      EMGMAX=0.0
      DO 20 J=1,NPTS
      DATUMS(J)=(DATUMS(J)-OFFSET)/EMGCAL
      IF(ABS(DATUMS(J)).GT.EMGMAX) EMGMAX=DATUMS(J)
   20 CONTINUE
C
C       FIND INTEGRATED EMG VALUE
C
      NPTS=NSECS*1024-1
      H=1./FLOAT(IRATE)
      SUMO=0.0
      SUME=0.0
      NPTS2=NPTS-2
      DO 40 I=1,NPTS2,2
      SUME=SUME+ABS(DATUMS(I+1))
      SUMO=SUMO+ABS(DATUMS(I))
   40 CONTINUE
      VOLTSEC=(2.0*SUMO+4.0*SUME-ABS(DATUMS(1))+ABS(DATUMS(NPTS)))*H/3.0
C
C       CALCULATE THE VARIANCE OF THE DATA
C
      EMGVAR=0.0
      DO 50 J=1,NPTS
      EMGVAR=EMGVAR+DATUMS(J)**2
   50 CONTINUE
      EMGVAR=EMGVAR/FLOAT(NPTS-1)
      RETURN
      END
      SUBROUTINE FORCE (DTYPE,NSECS,SPAN,FDGAIN)
C
C      SUBROUTINE FORCE SCALES ALL FORCE DATA FOR PLOTTING ON THE COMPUTEK A
C        AVERAGES FORCE DATA (OVER INTERVALS DETERMINED BY DATA CARD REQUEST)
C        FOR PRINTED OUTPUT.
C
C        CALLING ARGUMENTS:  DTYPE - TYPE OF DATA
C
C                            NSECS - LENGTH OF DATA SLICE
C
C                            SPAN - AVERAGING INTERVAL FOR PRINTED OUTPUT
C
C                            FDGAIN- AMOUNT OF GAIN WHICH WAS APPLIED TO FOR
C                                    DATA
C
      DIMENSION POUNDS(2600),AVGLBS(130)
      INTEGER DTYPE
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (POUNDS,DATUMS(1)),(AVGLBS,DATUMS(4000))
      EQUIVALENCE (FCGAIN,CAL(1,4)),(ZEROFOR,CAL(1,3))
      EQUIVALENCE (FORSLP,CAL(1,2)),(FORMAX,DATUMS(4500))
      GAIN=FCGAIN/FDGAIN
C
C      FIND ZERO BASELINE BEFORE BEGINNING OF FORCE DATA
C
      SUM=0.0
      DO 30 J=1,20
      SUM=SUM+POUNDS(J)
   30 CONTINUE
      BASE=SUM/20.
C
C      SCALE FORCE DATA INTO POUNDS
C
      FORMAX=0.0
      NPTS=20*NSECS
      DO 40 J=1,NPTS
      POUNDS(J)=(POUNDS(J)-BASE)*GAIN/FORSLP
      IF(POUNDS(J).GT.FORMAX) FORMAX=POUNDS(J)
   40 CONTINUE
C
C      AVERAGE FORCE DATA OVER SPAN INTERVAL
C
      LOOPS=NSECS/IFIX(SPAN)
      NPTS=20*IFIX(SPAN)
      DO 60 J=1,LOOPS
      SUM=0.0
      DO 50 K=1,NPTS
      SUM=SUM+POUNDS((J-1)*NPTS+K)
   50 CONTINUE
      AVGLBS(J)=SUM/FLOAT(NPTS)
   60 CONTINUE
      RETURN
      END
      OVERLAY OV6
      SUBROUTINE FFTPSD (NSECS,SPAN,PSDSUM,STDERR)
C
C      SUBROUTINE FFTPSD TAKES THE FOURIER TRANSFORM OF EMG DATA AND CALCULAT
C        THE RAW POWER SPECTRAL DENSITY UP TO 400 HZ. ACCORDING TO THE DATA
C        REQUEST A SMOOTHED PSD IS CALCULATED WHICH HAS IMPROVED STATISTICAL
C        PROPERTIES. TO REFLECT THOSE STATISTICAL PROPERTIES THE NORMALIZED
C        STANDARD ERROR IS ALSO CALCULATED.
C
C        CALLING ARGUMENTS: NSECS - LENGTH OF DATA SLICE
C
C                           SPAN   - BANDWIDTH OF SMOOTHED PSD
C
C                           PSDSUM- AREA UNDER POWER SPECTRUM
C
C                           STDERR- NORMALIZED STANDARD ERROR OF PSD
C
      DIMENSION PSD(2048),PSDN(410),PERCNT(410)
      DIMENSION CUMPCT(410),S(1024)
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (PSD,DATUMS(1))
      EQUIVALENCE (IRATE,IHEAD(11)),(PSDN,DATUMS(4000))
      EQUIVALENCE (PERCNT,DATUMS(3000)),(CUMPCT,DATUMS(3500))
      EQUIVALENCE (EXPVAL,DATUMS(2500)),(STDEV,DATUMS(2501))
      CONTINUE
C
C      APPLY COSINE WINDOW TO DATA
C
      CALL WINDOW (NSECS)
C
C      DETERMINE THE POWER OF 2 AND CORRESPONDING NUMBER OF DATA POINTS
C
      NEXP=10+NSECS/2
      IF(NEXP.EQ.14) NEXP=13
      N=2**NEXP
      PTS=FLOAT(N)
C
C      TAKE FFT OF THE DATA
C
      CALL RFFT (DATUMS,NEXP,S,-1,IFERR)
C
C      CALCULATE RAW POWER SPECTRAL DENSITY
C
      H=1./FLOAT(IRATE)
      K=1
      N1=N+1
      DO 10 J=3,N1,2
      PSD(K)=8.*H/PTS*(DATUMS(J)**2+DATUMS(J+1)**2)
      K=K+1
   10 CONTINUE
C
C      CORRECT PSD AMPLITUDES FOR WINDOW REDUCTION
C
      N2=N/2
      DO 15 J=1,N2
      PSD(J)=1.1429*PSD(J)
   15 CONTINUE
C
C      OUTPUT RAW PSD TO TAPE
C
      CALL PSDSAVE (NSECS)
C
```

```
C       CALCULATE SMOOTHED PSD
C
      F0=1./(PTS/FLOAT(IRATE))
      IOTA=IFIX(SPAN/F0)
      SPAN=FLOAT(IOTA)*F0
      PSDMAX=0.0
      K=1
      N2=N2/IOTA*IOTA
      DO 30 J=1,N2,IOTA
      TEMP=0.0
      DO 20 JJ=1,IOTA
      TEMP=TEMP+PSD(J+JJ-1)
   20 CONTINUE
      PSD(K)=TEMP/FLOAT(IOTA)
      IF(FLOAT(J/IOTA)*SPAN.GT.400.+SPAN) GO TO 25
      IF(PSD(K).GT.PSDMAX) PSDMAX=PSD(K)
   25 K=K+1
   30 CONTINUE
C
C       CALCULATE THE NORMALIZED PSD
C
      LIM=IFIX(400./SPAN)+1
      DO 40 J=1,LIM
      PSDN(J)=PSD(J)/PSDMAX
   40 CONTINUE
C
C       INTEGRATE THE SMOOTHED PSD
C
      PSDSUM=0.0
      DO 50 J=1,LIM
      PSDSUM=PSDSUM+SPAN*PSD(J)
   50 CONTINUE
C
C       CALCULATE % EACH BANDWIDTH IS OF TOTAL AND FIND THE CUMLATIVE % OF TOT
C
      DO 60 J=1,LIM
      PERCNT(J)=SPAN*PSD(J)/PSDSUM*100.
      IF(J.NE.1) GO TO 55
      CUMPCT(1)=PERCNT(1)
      GO TO 60
   55 CUMPCT(J)=CUMPCT(J-1)+PERCNT(J)
   60 CONTINUE
C
C       CALCULATE THE EXPECTED VALUE AND VARIANCE OF THE PSD
C
      F=SPAN/2.
      EXPVAL=0.0
      DO 70 J=1,LIM
      EXPVAL=EXPVAL+(PERCNT(J)/100.)*F
      F=F+SPAN
   70 CONTINUE
      F=SPAN/2.
      VARVAL=0.0
      DO 80 J=1,LIM
      VARVAL=VARVAL+(PERCNT(J)/100.)*(F-EXPVAL)**2
      F=F+SPAN
   80 CONTINUE
      STDEV=SQRT(VARVAL)
C
C       FIND THE NORMALIZED STANDARD ERROR
C
      STDERR=SQRT(1./FLOAT(IOTA))
      RETURN
```

END

```
      SUBROUTINE WINDOW (NSECS)
C
C      SUBROUTINE WINDOW APPLIES A COSINE TAPER TO THE FIRST AND LAST TENTHS
C        THE DATA TO REDUCE LEAKAGE.
C
C        CALLING ARGUMENTS:  NSECS - LENGTH OF THE DATA
C
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      PI=3.1415927
      NPTS=1024*NSECS
      IEDGE=NPTS/10
      TTOTAL=1.024*FLOAT(NSECS)
      K=1
      DO 10 J=1,IEDGE
      T1=.001*FLOAT(J)
      C1=.5*(1.-COS(PI*T1/(.1*TTOTAL)))
      DATUMS(J)=C1*DATUMS(J)
      DATUMS(NPTS-J+1)=C1*DATUMS(NPTS-J+1)
   10 CONTINUE
      RETURN
      END
      SUBROUTINE RFFT(A,M,S,IFS,IFERR)
C     ONE-DIMENSIONAL REAL FINITE FOURIER TRANSFORM
C
C
C     FOURIER TRANSFORM SUBROUTINE FOR REAL DATA.
C     PROGRAMMED IN SYSTEM/360, BASIC PROGRAMMING SUPPORT,
C     FORTRAN IV, (SEE FORM  C28-6504).
C     THIS DECK IS SET UP FOR IBSYS ON THE IBM 7094
C
C     THIS PROGRAM USES THE SUBROUTINE FFT TO COMPUTE COMPLEX
C     FOURIER TRANSFORMS OF REAL DATA.  PK FORT  S.D.A. NO. 3465 IS
C     AVAILABLE THROUGH SHARE.
C
C     THE FOURIER SERIES IS
C
C     X(J)= SUM OVER K=0 TO N, OF C(K)*EXP(2*PI*I*J*K/N )
C     J=0,1,2,...,N-1
C
C     WHERE I=SQRT(-1) AND WHERE C(K) IS COMPLEX.
C     SINCE X(J) IS REAL, C(K)= CONJG(C(N-K) ). THEREFORE ONLY
C     C(K),K= 0,1,...,N/2 ARE COMPUTED AND/OR USED.
C
C     ARGUMENTS-
C
C     A IS INITIALLY THE INPUT ARRAY, X, WHEN COMPUTING A FOURIER
C     TRANSFORM AND C WHEN COMPUTING A FOURIER SERIES. A IS REPLACES BY
C     THE OUTPUT ARRAY, C IN THE FORMER CASE, X IN THE LATTER.
C     THE X VECTOR CONTAINS THE REAL DATA X(0),X(1),...,X(N-1)
C     THE C VECTOR CONTAINS THE COMPLEX FOURIER AMPLITUDES
C     C(0),C(1),...,C(N/2). THE COMPLEX VECTOR C IS STORED ACCORDING
C     TO THE NORMAL FORTRAN IV CONVENTION FOR STORING COMPLEX NUMBERS.
C     I.E., REAL PARTS IN ALTERNATE CELLS STARTING WITH THE FIRST,
C     IMAGINARY PARTS IN ALTERNATE CELLS STARTING WITH THE SECOND.
C     TO ADHERE TO FORTRAN RULES, X(0), X(1),..., ARE REFERRED TO AS
C     X(1),X(2),..., RESP. IN THE PROGRAMS.  ALSO, C(0),C(1),... ARE
C     REFERRED TO AS C(1),C(2),..., RESP., IF C IS DESIGNATED AS
C     COMPLEX IN A TYPE STATEMENT.
C
C     M GIVES N=2**M
C
C     THE ARGUMENTS S, IFS, AND IFERR ARE THE SAME AS IN THE
C     SUBROUTINE FFT AND THE USER IS REFERRED TO THE COMMENT CARDS
C     IN FFT FOR THEIR EXPLANATION.
C     DIMENSION STATEMENTS- THE DIMENSIONS OF ARRAYS A AND S SHOULD
C     BE N+2 AND N/4, RESP. FOR THE LARGEST N TO BE USED. FOR
C     EXAMPLE, IF THE LARGEST M IS 13, THEN, N=8192 AND ONE SHOULD
C     HAVE THE DIMENSION STATEMENT-
C     DIMENSION A(8194), S(2048)
C     IF ONE WISHES TO SPECIFY A TO BE COMPLEX BY A TYPE STATEMENT,
C     ONE SHOULD GIVE IT A DIMENSION OF N/2 +1 , FOR THE LARGEST N.
C
C
      DIMENSION A(4500),S(1024)
      IFERRS = 0
      N=2**M
      NV2 = N / 2
      NV4M1 = N/4 - 1
      MM1 = M - 1
      IF (IABS(IFS)-1) 40,40,20
   20 IF (MP-M) 30,50,50
   30 IFERRS = 1
   40 NP = N
      MP = M
      CALL FFT (A,M,S,0,IFERR1)
      IFERRS = IFERRS + IFERR1
   50 KD = NP / N
      KT = KD
      NPV4 = NP / 4
      IF (IFS) 60,80,90
   60 CALL FFT(A,MM1,S,-2,IFERR2)
      IFERRS = IFERRS + IFERR2
      DO 70 K=1,NV4M1
      J=NV2-K
      A1R= A(2*K+1) + A(2*J+1)
      A1I=A(2*K+2)-A(2*J+2)
      A2R=A(2*K+2)+A(2*J+2)
      A2I=A(2*J+1)-A(2*K+1)
      KKT = NPV4-KT
      AWR=A2R*S(KKT) + A2I*S(KT)
      AWI = A2I*S(KKT)- A2R*S(KT)
      A(2*K+1)=(A1R+AWR)/4.
      A(2*K+2)=(A1I+AWI)/4.
      A(2*J+1)=(A1R-AWR)/4.
      A(2*J+2)=(AWI-A1I)/4.
   70 KT=KT+KD
      T=A(1)
      A(1)=(T+A(2))/2.
      A(N+1) = (T-A(2))/2.
      A(2)=0.
      A(N+2) = 0.
      A(NV2+1) = .5*A(NV2+1)
      A(NV2+2) = -.5 * A(NV2+2)
   80 IFERR = IFERRS
      RETURN
   90 DO 100 K=1,NV4M1
      J=NV2-K
      A1R=A(2*K+1) + A(2*J+1)
      A1I=A(2*K+2)-A(2*J+2)
      AWR=A(2*K+1)-A(2*J+1)
      AWI=A(2*K+2)+A(2*J+2)
      KKT = NPV4 - KT
      A2R=AWR*S(KKT) - AWI*S(KT)
      A2I=AWR*S(KT) + AWI*S(KKT)
      A(2*K+1) = A1R - A2I
      A(2*K+2) = A1I + A2R
      A(2*J+1) = A1R + A2I
      A(2*J+2) = A2R - A1I
  100 KT = KT + KD
      T = A(1)
      A(1) = T + A(N+1)
      A(2) = T - A(N+1)
      A(NV2+1) = 2.*A(NV2+1)
      A(NV2+2) =-2.*A(NV2+2)
      CALL FFT(A,MM1,S,2,IFERR2)
      IFERRS = IFERRS+IFERR2
      GO TO 80
      END
      SUBROUTINE FFT(A,M,S,IFS,IFERR)
C         FORT, ONE-DIMENSIONAL FINITE COMPLEX FOURIER TRANSFORM.
C
C
C     FOURIER TRANSFORM SUBROUTINE, PROGRAMMED IN SYSTEM/360,
C     BASIC PROGRAMMING SUPPORT, FORTRAN IV, FORM C28-6504
C     THIS DECK SET UP FOR IBSYS ON IBM 7094.
C
C     DOES EITHER FOURIER SYNTHESIS,I.E.,COMPUTES COMPLEX FOURIER SERIES
C     GIVEN A VECTOR OF N COMPLEX FOURIER AMPLITUDES,OR, GIVEN A VECTOR
C     OF COMPLEX DATA X DOES FOURIER ANALYSIS, COMPUTING AMPLITUDES.
C     A IS A COMPLEX VECTOR OF LENGTH N=2**M COMPLEX NOS. OR 2*N REAL
C     NUMBERS. A IS TO BE SET BY USER.
C     M  IS AN INTEGER 0.LT.M.LE.13, SET BY USER.
C     S IS A VECTOR S(J)= SIN(2*PI*J/NP ), J=1,2,....,NP/4-1,
C     COMPUTED BY PROGRAM.
C     IFS IS A PARAMETER TO BE SET BY USER AS FOLLOWS-
C     IFS=0 TO SET NP=2**M AND SET UP SINE TABLE.
C     IFS=1 TO SET N=NP=2**M, SET UP SIN TABLE, AND DO FOURIER
C     SYNTHESIS, REPLACING THE VECTOR A BY
C
C     X(J)= SUM OVER K=0,N-1 OF A(K)*EXP(2*PI*I/N)**(J*K),
C     J=0,N-1, WHERE I=SQRT(-1)
C
C     THE  X'S  ARE STORED WITH   RE X(J) IN CELL  2*J+1
C     AND  IM X(J)  IN CELL  2*J+2  FOR  J=0,1,2,...,N-1.
C     THE  A'S  ARE STORED IN THE SAME MANNER.
C
C     IFS=-1   TO SET  N=NP=2**M,SET UP SIN TABLE, AND DO FOURIER
C     ANALYSIS, TAKING THE INPUT VECTOR A AS X AND
C     REPLACING IT BY THE A  SATISFYING THE ABOVE FOURIER SERIES.
C     IFS>0 - SET UP SIN TABLE AND RETURN
C     IFS=+2 TO DO FOURIER SYNTHESIS ONLY, WITH A PRE-COMPUTED S.
C     IFS=-2 TO DO FOURIER ANALYSIS ONLY, WITH A PRE-COMPUTED S.
C
C     IFERR   IS SET BY PROGRAM TO-
C     =0 IF NO ERROR DETECTED.
C     =1 IF M IS OUT OF RANGE., OR, WHEN IFS=+2,-2, THE
C     PRE-COMPUTED S TABLE IS NOT LARGE ENOUGH.
C     =-1 WHEN IFS =+1,-1, MEANS ONE IS RECOMPUTING S TABLE
C     UNNECESSARILY.
C
C     NOTE- AS STATED ABOVE, THE MAXIMUM VALUE OF M FOR THIS PROGRAM
C     ON THE IBM 7094 IS 13. FOR 360 MACHINES HAVING GREATER STORAGE
C     CAPACITY, ONE MAY INCREASE THIS LIMIT BY REPLACING 13 IN
C     STATEMENT 3 BELOW BY LOG2 N, WHERE N IS THE MAX. NO. OF
C     COMPLEX NUMBERS ONE CAN STORE IN HIGH-SPEED CORE.  ONE MUST
C     ALSO ADD MORE  DO  STATEMENTS TO THE BINARY SORT ROUTINE
C     FOLLOWING STATEMENT  24  AND CHANGE THE EQUIVALENCE STATEMENTS
C     FOR THE  K'S.
C
      DIMENSION A(4500),S(1024),K(15)
      IF (M) 20,20,10
   10 IF (M-15) 40,40,20
   20 IFERR=1
   30 RETURN
   40 IFERR=0
      N=2**M
      IF (IABS(IFS)-1) 440,440,50
C     WE ARE DOING TRANSFORM ONLY. SEE IF PRE-COMPUTED
C     S TABLE IS SUFFICIENTLY LARGE
   50 IF (N-NP) 70,70,60
   60 IFERR=1
```

```
      GO TO 440
C     SCRAMBLE A, BY SANDE'S METHOD
   70 K(1)=2*N
      DO 80 L=2,M
   80 K(L)=K(L-1)/2
      DO 90 L=M,14
   90 K(L+1)=2
C      THE FOLLOWING 15 STATEMENTS ARE TO COMPENSATE FOR A WEAKNESS IN
C     THE FORTRAN V COMPILER
      K1 =K(15)
      K2 =K(14)
      K3 =K(13)
      K4 =K(12)
      K5 =K(11)
      K6 =K(10)
      K7 =K(9)
      K8 =K(8)
      K9 =K(7)
      K10=K(6)
      K11=K(5)
      K12=K(4)
      K13=K(3)
      K14=K(2)
      K15=K(1)
      N2 =K(1)
C     NOTE EQUIVALENCE OF KL AND K(14-L)
C     BINARY SORT-
      IJ =2
      J1 =2
  110 J2 =J1
  120 J3 =J2
  130 J4 =J3
  140 J5 =J4
  150 J6 =J5
  160 J7 =J6
  170 J8 =J7
  180 J9 =J8
  190 J10=J9
  200 J11=J10
  210 J12=J11
  220 J13=J12
  230 J14=J13
  240 JI=J14
  250 IF (IJ-JI) 260,270,270
  260 T=A(IJ-1 )
      A(IJ-1)=A(JI-1)
      A(JI-1)=T
      T=A(IJ)
      A(IJ)=A(JI)
      A(JI)=T
  270 IJ=IJ+2
      JI=JI+K14
      IF (JI.LE.K15) GO TO 250
      J14=J14+K13
      IF (J14.LE.K14) GO TO 240
      J13=J13+K12
      IF (J13.LE.K13) GO TO 230
      J12=J12+K11
      IF (J12.LE.K12) GO TO 220
      J11=J11+K10
      IF (J11.LE.K11) GO TO 210
      J10=J10+K9
      IF (J10.LE.K10) GO TO 200
      J9 =J9+K8
      IF (J9.LE.K9) GO TO 190
      J8=J8+K7
      IF (J8.LE.K8) GO TO 180
      J7=J7+K6
      IF (J7.LE.K7) GO TO 170
      J6=J6+K5
      IF (J6.LE.K6) GO TO 160
      J5=J5+K4
      IF (J5.LE.K5) GO TO 150
      J4=J4+K3
      IF (J4.LE.K4) GO TO 140
      J3=J3+K2
      IF (J3.LE.K3) GO TO 130
      J2=J2+K1
      IF (J2,LE.K2) GO TO 120
      J1=J1+2
      IF (J1.LE.K1) GO TO 110
      IF (IFS) 280,20,300
C     DOING FOURIER ANALYSIS,SO DIV. BY N AND CONJUGATE.
  280 FN = N
      DO 290 I=1,N
      A(2*I-1)=A(2*I-1)
  290 A(2*I)=-A(2*I)
C     SPECIAL CASE- L=1
  300 DO 310 I=1,N,2
      T = A(2*I-1)
      A(2*I-1) =T + A(2*I+1)
      A(2*I+1)=T-A(2*I+1)
      T=A(2*I)
      A(2*I) = T + A(2*I+2)
  310 A(2*I+2)= T - A(2*I+2)
      IF (M-1) 20,30,320
C     SET FOR L=2
  320 LEXP1=2
C     LEXP1=2**(L-1)
      LEXP=8
C     LEXP=2**(L+1)
      NPL= 2**MT
C     NPL = NP* 2**-L
      DO 390 L=2,M
C     SPECIAL CASE- J=0
      DO 340 I=2,N2,LEXP
      I1=I + LEXP1
      I2=I1+ LEXP1
      I3 =I2+LEXP1
      T=A(I-1)
      A(I-1) = T +A(I2-1)
      A(I2-1) = T-A(I2-1)
      T =A(I)
      A(I) = T+A(I2)
      A(I2) = T-A(I2)
      T= -A(I3)
      TI = A(I3-1)
      A(I3-1) = A(I1-1) - T
      A(I3   ) = A(I1 )    - TI
      A(I1-1) = A(I1-1) +T
  340 A(I1)    = A(I1    )  +TI
      IF (L-2) 380,380,350
  350 KLAST=N2-LEXP
      JJ=NPL
      DO 370 J=4,LEXP1,2
      NPJJ=N1-JJ
      UR=S(NPJJ)
      UI=S(JJ)
      ILAST=J+KLAST
      DO 360 I=J,ILAST,LEXP
      I1=I+LEXP1
      I2=I1+LEXP1
      I3=I2+LEXP1
      T=A(I2-1)*UR-A(I2)*UI
      TI=A(I2-1)*UI+A(I2)*UR
      A(I2-1)=A(I-1)-T
      A(I2  )=A(I   ) - TI
      A(I-1) =A(I-1)+T
      A(I)   =A(I)+TI
      T=-A(I3-1)*UI-A(I3)*UR
      TI=A(I3-1)*UR-A(I3)*UI
      A(I3-1)=A(I1-1)-T
      A(I3)   =A(I1  )-TI
      A(I1-1)=A(I1-1)+T
  360 A(I1)   =A(I1)    +TI
C     END OF I LOOP
  370 JJ=JJ+NPL
C     END OF J LOOP
  380 LEXP1=2*LEXP1
      LEXP = 2*LEXP
  390 NPL=NPL/2
C     END OF L LOOP
      IF (IFS) 410,20,30
C     DOING FOURIER ANALYSIS. REPLACE A BY CONJUGATE.
  410 DO 420 I=1,N
  420 A(2*I) =-A(2*I)
      GO TO 30
C     RETURN
C     MAKE TABLE OF S(J)=SIN(2*PI*J/NP),J=1,2,....NT-1,NT=NP/4
  440 NP=N
      MP=M
      NT=N/4
      MT=M-2
      IF (MT) 510,510,450
  450 THETA=.7853981634
C     THETA=PI/2**(L+1)     FOR L=1
      JSTEP = NT
C     JSTEP = 2**( MT-L+1 ) FOR L=1
      JDIF = NT/2
C     JDIF = 2**(MT-L)  FOR L=1
      S(JDIF) = SIN(THETA)
      IF (MT-2) 510,470,470
  470 DO 500 L=2,MT
      THETA = THETA/2.
      JSTEP2 = JSTEP
      JSTEP = JDIF
      JDIF = JDIF/2
      S(JDIF)=SIN(THETA)
      JC1=NT-JDIF
      S(JC1)=COS(THETA)
      JLAST=NT-JSTEP2
      IF (JLAST-JSTEP) 500,480,480
  480 DO 490 J=JSTEP,JLAST,JSTEP
      JC=NT-J
      JD=J+JDIF
  490 S(JD)=S(J)*S(JC1)+S(JDIF)*S(JC)
  500 CONTINUE
  510 IF (IFS) 70,30,70
      END
      SUBROUTINE PSDSAVE (NSECS)
C
C       SUBROUTINE PSDSAVE OUTPUTS TO TAPE THE RAW PSD (1024 REAL NUMBERS PER
C        2048 WORD RECORD) PRECEDED BY A 25 WORD HEADER RECORD.
C
C        CALLING ARGUMENT:    NSECS - LENGTH OF TIME SERIES DATA SLICE
C
      DIMENSION IARRAY(2048)
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (IARRAY,DATUMS(2100))
      MTO=2
      IEOF=60000K
      IRITE=50000K
      IBFILE=40000K
C
C       CREATE AN OUTPUT HEADER RECORD
C
      DO 10 J=1,25
      IARRAY(J)=0
   10 CONTINUE
      DO 20 J=1,20
      IARRAY(J)=IHEAD(J)
   20 CONTINUE
      NRECS=NSECS/2
      IF(NRECS.EQ.0) NRECS=1
      IARRAY(21)=NRECS
      IWRD=IRITE+25
      CALL MTDIO (MTO,IWRD,IARRAY,ISTAT,IER)
C
C     OUTPUT FOLLOWING DATA RECORDS CONTAINING THE RAW PSD VALUES
C
      K=0
      DO 50 J=1,NRECS
      DO 40 JJ=1,1024
      DATUMS(2099+JJ)=DATUMS(K+JJ)
   40 CONTINUE
      IWRD=IRITE+2048
      CALL MTDIO (MTO,IWRD,IARRAY,ISTAT,IER)
      K=K+1024
   50 CONTINUE
C
C       WRITE TWO EOFS AND BACK UP TO JUST BEFORE THEM
C
      DO 60 J=1,2
   60 CALL MTDIO (MTO,IEOF,IARRAY,ISTAT,IER)
      IBACK=IBFILE+1
      DO 70 J=1,2
   70 CALL MTDIO (MTO,IBACK,IARRAY,ISTAT,IER)
      RETURN
      END
      OVERLAY OV7
      SUBROUTINE POUT(DTYPE,START,NSECS,SPAN,VOLTSEC,PSDSUM,STDERR,WHOA)
C
C      SUBROUTINE PLOT OUTPUTS TO THE LINE PRINTER HEADER AND CALIBRATION DAT
C       AVERAGED FORCE DATA, AND SMOOTHED PSD SPECTRUM.
C
C       CALLING ARGUMENTS:   DTYPE - TYPE OF DATA
C
C                            START - BEGIN TIME OF PRESENT DATA SLICE
C
C                            NSECS - LENGTH OF DATA SLICE
C
C                            SPAN  - FOR FORCE DATA NUMBER OF SECS FORCE DATA
C                                    AVERAGED OVER; FOR PSD DATA FREQUENCY
C                                    INTERVAL FOR SMOOTHING ESTIMATES
C
C                            VOLTSEC- INTEGRATED EMG VALUE (TIME DOMAIN)
C
C                            PSDSUM- INTEGRATED EMG VALUE (FREQUENCY DOMAIN)
C
C                            STDERR- NORMALIZED STANDARD ERROR OF ESTIMATE F
C                                    PSD VALUES
C
C                            WHOA  - FLAG SET WHEN DATA PROCESSING ATTEMPTED
C                                    WITHOUT CALIBRATION
C
C
      INTEGER DTYPE,WHOA
      DIMENSION IEDATE(3),IDDATE(3),CUMPCT(410)
      DIMENSION AVGLBS(130),PSD(410),ITIME(3),PSDN(410),PERCNT(410)
      COMMON DATUMS(4500),IHEAD(20),CAL(2,4),IVARSW(10)
      COMMON/IGD/ MUSCLE(7,4),IFORR(2)
      EQUIVALENCE (ISUBNO,IHEAD(1)),(IEDATE,IHEAD(2))
      EQUIVALENCE (IDDATE,IHEAD(5)),(IFLITE,IHEAD(8))
      EQUIVALENCE (IRUN,IHEAD(9)),(MUS,IHEAD(20))
      EQUIVALENCE (IATAPE,IHEAD(10)),(ISAMPE,IHEAD(11))
      EQUIVALENCE (ISAMPM,IHEAD(12)),(AVGLBS,DATUMS(4000))
      EQUIVALENCE (PSD,DATUMS(1)),(PSDN,DATUMS(4000))
      EQUIVALENCE (PERCNT,DATUMS(3000)),(CUMPCT,DATUMS(3500))
      EQUIVALENCE (EXPVAL,DATUMS(2500)),(STDEV,DATUMS(2501))
      EQUIVALENCE (EMGVAR,DATUMS(4500)),(ITIME,IHEAD(16))
      DATA MUSCLE /'BR','AC','HI','AL',' B','IC','EP',' B','. ','RA','DI
     @','AL','IS','  ',' G','AS','TR','OC','NE','MI','US',' S','OL','EU'
     @,'S ','  ','  ','  '/
      DATA IFORR /' F','R+'/
      LPT=12
      IF(WHOA.EQ.1) GO TO 150
C
C      BRANCH TO THE PROPER PRINT OUT SECTION FOR THE DATA TYPE
C
      IF(DTYPE-4) 5,50,100
C
C      PRINT OUT HEADER INFORMATION
C
    5 IP=1
      IF (IFLITE.GE.0) IP=2
      WRITE(LPT,10)
   10 FORMAT(1H1,//,48X,'CARDIOVASCULAR LABORATORY',//,47X,'EMG DATA PRO
     @CESSING PROGRAM',/,47X,'------------------------------')
      WRITE(LPT,20)ISUBNO,IEDATE,IFORR(IP),IFLITE,IRUN,IATAPE,IDDATE,(MU
     @SCLE(J,MUS),J=1,7),ISAMPE,ISAMPM
   20 FORMAT(////,47X,'*** HEADER INFORMATION***',///,' SUBJECT NO.:', 7
     @X,I3,18X,'EXPERIMENT DATE:',6X,I2,'/',I2,'/',I2,10X,'FLIGHT REFERE
   @NCE DAY:',A3,I3,//,' RUN NO.:',11X,I2,19X,'ANALOG TAPE NO.:', 7X,I
   @3,14X,'DIGITIZING DATE:',6X,I2,'/',I2,'/',I2,//,' MUSCLE:', 9X,7A2
   @,//,' EMG SAMPLE RATE (SAMP/SEC):',I5,8X,'FORCE SAMPLE RATE (SAMP/
   @SEC):',I5)
    WRITE(LPT,30) (CAL(1,J),J=1,4)
 30 FORMAT(////,49X,'** CALIBRATION DATA **',///,' FORCE CAL',/,' ----
   @------',//,' CAL WEIGHT (LBS):',F7.1,16X,'COUNTS/LB:',2X,F6.0,5X,'A
   @VG. BASELINE COUNT:',2X,F6.0,5X,'CAL GAIN:',2X,F3.0)
    WRITE(LPT,40) (CAL(2,J),J=1,2)
 40 FORMAT(///,' EMG CAL',/,' -------',//,' RMS AMPLITUDE (MICROVOLTS)
   @:',F6.0,7X,' COUNTS/MICROVOLT:',F5.2,//)
    GO TO 200
C
C     PRINT OUT FORCE DATA
C
 50 WRITE(LPT,60) ITIME,NSECS,SPAN
 60 FORMAT(1H1,51X,'*** FORCE DATA ***'///5X,'START TIME-',I3,':',I2,'
   @:',I2,15X,'DATA LENGTH (SECS)-',I4,15X,'AVERAGE FORCE INTERVAL (SE
   @CS)-',F3.0,//)
    LOOPS=NSECS/IFIX(SPAN)
    IF(LOOPS.GT.49) GO TO 66
    WRITE(LPT,62)
 62 FORMAT(47X,'INTERVAL',5X,'AVERAGE FORCE (LBS)'/,47X,'--------',5X,
   @'-------------------',//)
    DO 65 J=1,LOOPS
    WRITE(LPT,64) J,AVGLBS(J)
 64 FORMAT(49X,I3,15X,F5.1)
 65 CONTINUE
    GO TO 200
 66 IF((LOOPS/2)*2.NE.LOOPS) LOOPS=LOOPS-1
    WRITE(LPT ,68)
 68 FORMAT(15X,'INTERVAL',5X,'AVERAGE FORCE (LBS)',20X,'INTERVAL',5X,'
   @AVERAGE FORCE (LBS)',/,15X,'--------',5X,'-------------------',20X
   @,'--------',5X,'-------------------',//)
    LOOPS2=LOOPS/2
    DO 75 J=1,LOOPS2
    JJ=J+LOOPS2
    WRITE(LPT,73) J,AVGLBS(J),JJ,AVGLBS(JJ)
 73 FORMAT(17X,I3,15X,F5.1,29X,I3,15X,F5.1)
 75 CONTINUE
    GO TO 200
C
C     PRINT OUT PSD DATA
C
100 WRITE(LPT,110) ITIME,NSECS,VOLTSEC,SPAN,STDERR,PSDSUM,EXPVAL,STDEV
   @,EMGVAR
110 FORMAT(1H1,39X,'*** POWER SPECTRAL DENSITY OF EMG DATA ***'///4X,'
   @START TIME-',I3,':',I2,':',I2,11X,'DATA LENGTH (SECS)-',I4,14X,'IN
   @TEGRATED EMG (MICROVOLT*SEC)-',E12.4//4X,'BANDWIDTH (HZ)-',F6.3,10
   @X,'NORMALIZED STANDARD ERROR-',F6.3,5X,'INTEGRATED PSD (MICROVOLT*
   @*2)-'      ,E13.4,//,4X,'MEAN (HZ)-',F6.1,15X,'STANDARD DEVIATION
   @(HZ)-',F6.1,8X,'EMG VARIANCE (MICROVOLT**2)-',E15.4,/)
    LOOPS=IFIX(400./SPAN)+1
    FREQ=SPAN/2.
    IF(LOOPS.GT.46) GO TO 130
    WRITE(LPT,115)
115 FORMAT(30X,' FREQ',13X,'PSD',11X,'PSD',10X,'% OF',10X,'CUM %'/,30X
   @,' (HZ)',8X,'(MMV**2/HZ)',8X,'NORM',9X,'TOTAL',9X,'TOTAL',/, 31X,'
   @----',8X,'-----------' ,8X,'----',9X,'-----',9X,'-----',/)
    DO 125 J=1,LOOPS
    WRITE(LPT,120) FREQ,PSD(J),PSDN(J),PERCNT(J),CUMPCT(J)
120 FORMAT(30X,F6.2,7X,F10.3,8X,F6.4,7X,F6.2,8X,F6.2)
    FREQ=FREQ+SPAN
```

```
  125 CONTINUE
      GO 10 200
  130 IF((LOOPS/2)*2.NE.LOOPS) LOOPS=LOOPS-1
      FREQ2=FLOAT(LCOPS/2)*SPAN+FREQ
      WRITE(LPT,135)
  135 FORMAT('  FREQ', 8X,'PSD', 9X,'PSD',8X,'% OF',6X,'CUM %',22X,'FREQ
     @', 9X,'PSD' , 8X,'PSD',8X,'% OF',6X,'CUM %'/2X,'(HZ)',4X,'(MMV**2/
     @HZ)',5X,'NORM',7X,'TOTAL',5X,'TOTAL',22X,'(HZ)',5X,'(MMV**2/HZ)',5
     @X,'NORM',6X,'TOTAL',6X,'TOTAL',/,'  ----',4X,'-----------',5X,'---
     @-',7X,'-----',5X,'-----',22X,'----',5X,'-----------',5X,'----',6X,
     @'-----',6X,'-----',/)
       LOOPS2=LOOPS/2
       DO 145 J=1,LOOPS2
       JJ=J+LOOPS2
       WRITE(LPT,140) FREQ,PSD(J),PSDN(J),PERCNT(J),CUMPCT(J),FREQ2,PSD(J
     @J),PSDN(JJ),PERCNT(JJ),CUMPCT(JJ)
  140 FORMAT(F7.2,3X,F10.3,5X,F6.4,5X,F6.2,5X,F6.2,20X,F6.2,4X,F10.3,5X,
     @F6.4,5X,F6.2,5X,F6.2)
      FREQ=FREQ+SPAN
      FREQ2=FREQ2+SPAN
  145 CONTINUE
      GO TO 200
C
C      ERROR MESSAGE
C
  150 WRITE(LPT,160)
  160 FORMAT(' ERROR...DATA ANALYSIS ATTEMPTED WITHOUT CALIBRATION FIRST
     @.  CHECK DATA CARD ORDER.')
      STOP
  200 RETURN
      END
```

```
EMGAN.SV          LOADED BY RLDR REV 03.01 AT 15:34:36 09/05/75
 .MAIN
          001620
          000,000  POUT  003477
          000,001  TINPU
                   FIND
                   OUT60 002304
          000,002  DCAL  001035
          000,003  GRAPH 000165
          000,004  EMG
                   FORCE 001167
          000,005  FFTPS
                   WINDO
                   RFFT
                   FFT
                   PSDSA 007540
          000,006  CINPU 001217
          011620
 I
 FREAD
 THREA
 RDFLD
 READL
 OPEN
 FOVLD
 FOPEN
 DFILW
 CLOSE
 FREDI
 FALOC
 ARYSZ
 FSBR
 CGT
 IABS
 SMPY
 SDVD
 IFIX
 ABSLT
 RLSE
 INIT
 MTDIO
 IPWER
 RIPWR
 COS
 SQRT
 PLY1
 BREAK
 FLIP
 ARGUM
 FRGLD
 FARGO
 FL
 STREG
 LDREG
 MVBT
 LDO
 STOP
 FINIT
 FLINK
 RTER
 WRCH
 BDASC
 BASC
```

```
COUT
LDSTB
MOVEF
CPYAR
MAD
FPZER
FPTRS
DUMMY
ARDUM
MMPYD
TMIN


     NMAX 023223
     ZMAX 000231
     CSZE 021526
      EST 000000
      SST 000000

    .FREA 000050
    .FWRI 000051
    .BRD  000052
    .BWR  000053
    .ALLO 000074
    .THRE 000075
    .RDFL 000076
    .RDFC 000077
    .WRIT 000100
    .READ 000101
    .WRTS 000102
    .REDS 000103
    .FOPE 000104
    .FRED 000105
    .FALU 000106
    .ARYS 000107
    .FSUB 000110
    .FSBR 000111
    .CGT  000112
    IA.S  000113
    .SMPY 000114
    .SDVD 000115
    XI.X  000116
    IF.X  000116
    .ABS  000117
    .IPWR 000120
    .FLIP 000127
    .FARG 000131
    .FRGL 000132
    .FRG0 000133
    .FRG1 000134
    SN.L  000135
    DB.E  000135
    FL.AT 000143
    .MVBC 000160
    .MVBT 000161
    .LD0  000162
    .LD1  000163
    .LD2  000164
    .ST0  000165
    .ST1  000166
    .ST2  000167
    .STOP 000170
    .PAUS 000171
```

```
  .FINI 000172
  .FCAL 000173
  .FSAV 000175
  .FRET 000176
  .RTER 000200
  .RTEO 000201
  .RTES 000202
  .WRCH 000203
  .COUT 000204
  .CIN  000205
  .LDBT 000206
  .STBT 000207
  .MOVE 000210
  .CPYA 000211
  .CPYL 000212
  .MAD  000213
  .MADO 000214
  .IOCA 000215
  SUCOM 000216
  .NDSP 000217
  TVR   000217
  AFSE  000220
  SP    000221
  .OVFL 000222
  .SVO  000223
  GSP   000224
  NSP   000225
  FLSP  000225
  USTAD 000400
C IGD   000444 000036
  .MAIN 000602
  EMG   001710
  CINPU 001716
  DCAL  001727
  TINPU 001730
  GRAPH 001753
  FFTPS 001753
  POUT  002016
  FSAV  002175
  FRET  002176
  FQRET 002177
  FORCE 002462
  FIND  003147
  WINDO 003225
  RFFT  003450
  OUT60 003612
  FFT   005333
  XAS.  006117
  DABS. 006117
  ABS.  006117
  FIPR1 006121
  COS.  006122
  SIN.  006123
  SQRT. 006124
  FPLY1 006125
  FBRK1 006126
  FLIP2 006130
  FLIP1 006130
  FFLD1 006135
  FFST1 006136
  FAD1  006137
  FSB1  006140
```

```
     FDV1  006142
     FXFL1 006143
     FLFX1 006144
     FSGN1 006145
     FNEG1 006146
     FCLE1 006147
     FCLT1 006150
  2  FCGE1 006151
     FCGT1 006152
     FCEQ1 006153
     FRST1 006154
     FRST2 006155
     FRLD2 006156
     FRLD1 006157
     FCALL 006173
     FRCAL 006174
     MPY0  006226
     MPY   006227
     DVD   006230
     PSDSA 011007
     .I    011656
     IOPTR 012024
     FERT0 012052
     FERT1 012055
     FERTN 012061
     BRD   012100
     BWR   012104
     FREAD 012111
     FWRIT 012115
     ALLOC 015761
     THREA 016010
     RDFLD 016025
     RDFCH 016031
     READL 016142
     WRITL 016152
     REDS  016366
     WRITS 016377
     MTOPD 016460
     APPEN 016463
     OVOPN 016466
     OPEN  016471
     OVLOD 016740
     FOVLD 016740
     FOPEN 017007
     DFILW 017205
     DELET 017205
     CLOSE 017237
     FCLOS 017237
     FREDI 017307
     FALOC 017420
     ARYSZ 017454
     FSUBA 017472
     FSBR  017635
     CGT   017671
     .IABS 017714
     SMPY  017723
     SDVD  017747
     .IFIX 020017
     ABS   020030
     RLSE  020035
     INIT  020053
     MTDIO 020072
     IPWR  020146
```

```
RIPWR  020223
CS     020273
SN     020277
SWR    020440
PLY1   020600
BRK    020633
FLPO   020650
FLP    020653
FARGU  020676
FRGLD  020732
FRG1   020742
FRGO   020747
FL     020767
FS     021026
FB     021107
FA     021110
FM     021240
FXL    021377
FLX    021435
FSG    021502
FNG    021523
FLE    021557
FLT    021561
FGE    021563
FGT    021565
FEQ    021567
FD     021637
ST1    021742
ST2    021744
LDR1   021767
LDR2   021771
MVBT   022023
MVBC   022027
LD0    022060
LD1    022062
LD2    022064
EXIT   022100
STOP   022105
PAUSE  022112
FINIT  022163
SAV0   022205
SAV2   022255
SAV3   022262
RSTR   022275
QRSTR  022311
.OFLO  022313
RTER   022355
RTESP  022357
RTE0   022371
WRCH   022571
.BDAS  022606
.BASC  022670
COUT   022725
CIN    022737
LDB    022746
STB    022760
MOVEF  022776
CPYAR  023023
CPYLS  023054
MAD    023063
MADO   023064
.FLSP  023110
IMIN   023127
```

```
FHMA   177777
FRTSK  177777
.FLSZ  177777
QTCK   177777
OV7    000,000
OV2    000,001
OV3    000,002
OV4    000,003
OV5    000,004
OV6    000,005
OV1    000,006
```

APPENDIX H
CF-16/A GENERAL A/D PROGRAM DOCUMENTATION

Author: Winston Blackmon
Engineering Systems Branch
Institutional Data Systems Division
Data Systems and Analysis Directorate

4.2.0 LBNP ONLINE WITH SIGMA 3
REQUIREMENT DELETED

4.3.0.0 General Data Acquisition System.

The DBAS (General Data Acquisition System) acquires, digitizes, and records test data from an analog playback. The user is allowed up to 3 different rates and sixteen channels of input. The only restriction are that the rates must be integer multiples of each other and that the A/D channels be grouped in the order of highest rate, mid-rate, and slow rate. The GDAS is loaded from the nine-track tape and will operate independently of the Sigma 3 Biomedical System. It will, however, require the FM recorder and Systron Donner used by the Sigma 3 unless the user can supply his own recorder and Systron Donner. The GDAS data storage and output, operator interaction, and subroutines are described in the following sections.

4.3.1.0 Data Storage

There are two types of data storage - that which is used by the executive for control purposes; and that which is used as buffers for digitized data accumulation. The data used by the executive for control is stored in locations 80_{16} to $A5_{16}$ of scratch pad (locations $0\text{-}255_{10}$). The following table defines these data.

LOCATION	VARIABLE	DESCRIPTION
80_{16} (128_{10})	CLCNT	User input number of microseconds to elapse between each A/D read of the highest speed data.
81_{16} (129_{10})	NDRATE	User input number of different rates (1, 2, or 3)
82_{16} (130_{10})	NØLØ	Minus total number of channels (computed from user input)
83_{16} (131_{10})	LØCNT	Number of reads of the high speed channels before the lowest speed channel is read.
84_{16} (132_{10})	NØMED	User input minus total number of medium and high speed channels.
85_{16} (133_{10})	MEDCNT	Number of reads of the high speed channels before the medium speed channels are read (user input)
86_{16} (134_{10})	NØHS	Number of high speed channels to read - user input

LOCATION	VARIABLE	DESCRIPTION
87_{16} (135_{10})	BUFZER	Buffer zero starting address minus 1. Fixed at $77C_{16}$
88_{16} (136_{10})	BUFZND	Buffer zero end address. Set to $B64_{16}$, but will be changed if the user sets the data buffer length to less than 1000.
89_{16} (137_{10})	BUFØNE	Buffer one starting address minus 1. Fixed at $B7D_{16}$.
$8A_{16}$ (138_{10})	NDBUF1	End address for Buffer one. Set at $F65_{16}$, but will be changed if the user selects a data buffer length less than 1000.
$8B_{16}$ (139_{10})	BUFUSE	Buffer in use pointer. Initially set at zero but is changed when data buffer 1 is being filled with digitized data.
$8C_{16}$ (140_{10})	MAXBUF	Maximum buffer length including header information. Fixed at 401_{16} (1025_{10})
$8D_{16}$ (141_{10})	TPØUT	Tape output flag. Initially set to zero, but is set to 1 when a data buffer has been filled and is ready for output to tape.
$8E_{16}$ (142_{10})	DØNCLK	Flag, initially zero, set to one whenever the Systron Donner clock timer has been read and are ready to be unpacked.
$8F_{16}$ (143_{10})	BGNCHN	Address of the first high speed A/D convertor channel. Default value is zero.
90_{16} (144_{10})	ADCRD	Flag which indicates an A/D convertor read is in progress (= 1)
91_{16} (145_{10}) 92_{16} (146_{10}) 93_{16} (147_{10}) 94_{16} (148_{10})	DØNMS DNSEL DNMIN DNØUR	Timer read from the Systron Donner when data buffers are switched. These times are in a pseudo BCD format, Complemented.
95_{16} (149_{10}) 96_{16} (150_{10}) 97_{16} (151_{10}) 98_{16} (152_{10})	STHR STMIN STSEL STRMS	Systron Donner timer at which the user wants to begin digitizing data. Default values are zero.

LOCATION		VARIABLE	DESCRIPTION
99_{16}	(153_{10})	STPHR	Systron Donner timer at which the the user wants to stop digitizing data. Default values are $1E00_{16}$.
$9A_{16}$	(154_{10})	STPMIN	
$9B_{16}$	(155_{10})	STPSEL	
$9C_{16}$	(156_{10})	STØPMS	
$9D_{16}$	(157_{10})	BUFHD1	Buffer one header address minus one. Used for tape output.
$9E_{16}$	(158_{10})	BUFHDO	Buffer zero header address minus one. Used for tape output.
$9D_{16}$	(159_{10})	MBUFLG	Negative of Maximum buffer length including header. Set by user with a default value of -1025_{10}.
$A0_{16}$	(160_{10})	MXDBUF	Maximum data buffer length. Set at 1000_{10}.
$A1_{16}$	(161_{10})	PAR	Number of parity errors detected during tape output of the digitized data.
$A2_{16}$	(162_{10})	TIMG	Number of timing errors detected during tape output of the digitized data.
$A3_{16}$	(163_{10})	HDBFLG	Length of the header portion of each data buffer. Set at 25_{10}.
$A4_{16}$	(164_{10})	BUFTTY	Address of the teletype data buffer. Address is 100_{16} (256_{10}).
$A5_{16}$	(165_{10})	STATUS	Temporary storage for the status word of the tape write operation.
$A6_{16}$	(166_{10})	TAPEND	Flag signifying end of operation on tape. Writes during test.

Also used for control purposes, are the addresses of subroutines which are general in nature or called by more than one other subroutine. These are defined below

70_{16}	(112_{10})	FETCH	Used to coordinate data input, e.g., reads a buffer from teletype and checks for data errors.
71_{16}	(113_{10})	CRLF	Outputs a carriage return-line feed to to the teletype.
72_{16}	(114_{10})	ØTL	Outputs a line to the teletype.

$73_{16}(115_{10})$	UNPACK	Unpacks the System Donner Time from pseudo BCD to binary.
$74_{16}(116_{10})$	CØNVR	Converts ASCII Input number to a signed binary integer and checks for overflow (number too large).
$75_{16}(117_{10})$	TDUMP	Starts tape I/O during a test and writes EØF Marks after test.
$76_{16}(118_{10})$	ØDEL	Converts a binary integer to a string of ASCII digits for output.
$77_{16}(119_{10})$	ITT	Inputs one ASCII Character from the teletype.
$78_{16}(120_{10})$	ØTT	Outputs one ASCII character to the teletype.
$79_{16}(121_{10})$	TYPIN	Routine called to fill the teletype input buffer.
$7A_{16}(122_{10})$	RESTØR	Restores the executive data table (locations 80_{16} to $A5_{16}$) to it's initial value, and zero the digitized data buffer areas.
$7B_{16}(123_{10})$	STØPIT	Routine to stop the data acquisition and restore the acquisition routine to its initial state.

The buffers which are used to accumulate the digitized data consist of two blocks of core, each 1025_{10} words in length. The first 25_{10} words of each block are leader data, 20_{10} of which may be set by the user. The blocks are described below.

LOCATION	BUFFER WORD	PURPOSE
764_{16} $B65_{16}$	1	Buffer record counter incremental by the executive.
$765\text{-}768_{16}$ $B66\text{-}B69_{16}$	2 - 5	Storage for Systron Donner time at which the first data for this record were acquired.
$769\text{-}77C_{16}$ $B6A\text{-}B7D_{16}$	6-25	Header information stored by the user. The 20_{10} words are constant during a given acquisition.

$77D\text{-}B64_{16}$ $B7E\text{-}F65_{16}$ }	26 1025	Digitized data. Note that these end addresses are only valid for the maximum data buffer length, 1000_{10}. Also the data will be stored in the order that it is read, i.e. there is no attempt to group the data by channels, rates, etc.

4.3.2.0 Operator Interaction with the DGAS

The GDAS acts in a conversational manner with the user via the teletype. In response to questions and for all data (numeric) input, a slash (/) is used to terminate the input. For correction of input prior to the slash, the last character input can be deleted by an up arrow (↑) and the entire line deleted by a back (←). Of the particular input is rejected by the GDAS, the back arrow and a carriage return - line feed will be printed. The quantity will be requested again.

The following inputs are requested by the GDAS.

6 DAS. No. Rates $\left\{\begin{matrix}1\\2\\3\end{matrix}\right\}$	Number of different data rates, the user must input, 1, 2, or 3.
MICROSEC PER HI SPD READ.	Number of microseconds to elapse between reads of the highest data rate. For example, 1000 samples/second would be 1000 microseconds; 100 samples/second would be 10000 microseconds. The lowest rate which can be input for a high speed rate is 62500 microseconds/read or 16 samples/second. This does not preclude lower rates if more than one rate is used.
DATA BUF LENGTH, MAX I S 1000	Length of the data buffer to be filled before output to tape. This number should be chosen with care-especially with multiple data rates-so that the data from each rate will appear at the same place in each buffer. For example, with four channels of data at 320 samples/second and eight channels at 2 samples/second, a good number for buffer length would be 648. (4x160+8). The buffer would be dumped each ½ second.
ADC CHANNEL	Address of the first A/D convertor channel to be read, i.e. the first high speed channel.

GAIN	Gain to be applied to each A/D conversion. Gains available are times 1, 2, 4, or 8.
NO. HI SPD CHANS.	Number of high speed channels to read. For one rate, this number will be the total number of channels.
NO. MED SPED CHANS.	Number of medium speed channels to read. The output applies only when there are two or three rates. When there are two rates, this is the number of lower speed channels to read.
NO. LO SPED CHANS.	Number of low speed channels. This output applies only when there are three rates. This input is the number of channels for the lowest rate.
NO. HI SPD READS/MEDIUM READ.	Input the number of times to read the fastest rate before reading the mid-rate (or in the case of two rates, the low rate). e.g., Suppose the high rate is 100 samples/second and the mid rate is 25 samples/second. The number to be input would be 4 = 100/25.
NO. HISPD READS/LOW READ.	In the case of three data rates, how many times are the highest speed channels read before the lowest speed channels are read? Eg. Suppose the rates are 500, 250, and 25 samples per second, respectively. The response to this input would be 500/25 = 20.
INPUT DATA FOR TAPE HEADER?	Yes or no response. This question asks if the user desires to store constant numeric data in words 6 through 25 in the header of each buffer. This data is output to tape with the digitized data. See 4.3.1.0 for further data description. If a yes response is given, the GDAS asks for the input to word 6, then 7, then eight, etc. If the user desires to terminate the header input, he may enter letter A followed by a slash.
WRITE BEGINNING EØF?	This allows the user to write a end of file on the beginning of his tape. One reason to do so is to allow the tape to space past the extra 3 inches at the beginning. This 3 inch gap is especially important during high speed acquisition rates. (See 4.3.2.1 for more detail.)

START TIMES?	Does the user want to input the Systron Donner times (hr, min, Sec) at which the GDAS is to start digitizing data? Yes or No response required! If yes, the user must input the times as requested If no, the GDAS uses zero as the default value.
STØP TIMES?	Same as start times except these times are when to stop acquiring data. The default value is $1E00_{16}$ if a no response is entered.

A $\longrightarrow$

S TO START	This is an input which prevents the GDAS from looking at the start times. It is useful if the default is entered or if the user must position an analog tape for playback. The quantity S/ will start the GDAS.

When a test is completed, either by reaching the stop times or by an x input, the GDAS will allow additional tests to be conducted with the same data input. This question is

CONTINUE TEST WITH NEW TIMES?

If a yes response is entered, then new start/stop times are requested. Then the test may be restarted by using the s/input. If a no response is entered, the GDAS will ask of the tape is to be rewound between tests. If yes, it rewinds the tapes. In either case, the GDAS initializes itself, i.e., the user must enter new inputs for rates, etc.

4.3.2.1. Timing Estimates for GDAS

The GDAS will digitize data at various rates, write various length records depending on the user's inputs. For this reason, the user should put some "handle" on whether his specific rates and data buffer lengths are feasibile. As mentioned previously, the minimum single rate speed is 16 samples/second. To determine the maximum rate, several tests were conducted. The results of these tests indicate that the tape speed will be the limiting factor. The only control the user has over tape speed is to vary his buffer length. A longer buffer will take longer to fill than to dump in most cases. The limiting case is:

$$\text{Time to fill buffer with data in seconds} > 0.046 + 0.0001\ (\text{buffer length in words}).$$

The above inequality has been tested with 6 channels of data at 1000 samples/second using various buffer lengths. The minimum buffer length which worked was 690. The time required to fill the buffer and to dump the buffer in this case is identical. Note that the above inequality and test are based on the GDAS not being required to write a record from the BOT (beinning of tape) position. The first operation from BOT requires .197 seconds to simply start and stop the tape. Hence, if the user is dealing with high rates, he must not start the tape from BOT. A beginning dummy record or an end of file can be used to position past the BOT.

4.3.3.0.0	Subroutine Description

All routines are documented according to the pattern shown below:

.0	name
.1	purpose
.2	calling sequence
.3	software/software interfaces
.4	input data
.5	output data
.6	storage required
.7	description
.8	flow diagrams

4.3.3.1.0	BEGIN - Pre- and Post Test Control Routine
4.3.3.1.1	The purpose of this program is to control the flow of input data from the user and to insure that all required inputs are within tolerances. BEGIN also starts the test by calling the test routine.
4.3.3.1.2	Calling Sequence - JST *BEGIN Where BEGIN is a core location which contains the value 110_{16}.
4.3.3.1.3	Software/Software Interfaces BEGIN uses both external and internal subroutines. If no indication is given, the subroutine should be assumed to be external.
RESTØR	Subroutine called to restores all the GDAS variables that may be changed by the user to their default values. It also resets all the I/Ø devices.
CRLF	Subroutine called to issue a carriage return - line feed to the teletype.
ØTL	Subroutine called to output a message to the teletype.
FETCH	Subroutine called to output a message and return the numeric response in the A register. It skips forward one instruction if the numeric input was free from character errors.
TYPIN	Subroutine called to fill the 16 character teletype input buffer, BUFR.
CHKIT	Internal subroutine used to check for the no change condition in variables whose input is optional. The routine FETCH returns an error condition for this particular case when possibly no input error has been made.

EINS -	Subroutine called to input data pertinent to a one rate data acquisition.
ZWEI -	Subroutine called to input data pertinent to a two rate data acquisition.
DREI -	Subroutine called to input data pertinent to a three rate data acquisition
HEADER	Subroutine called to input and store constant data into the 25 word tape buffer header. Only twenty words of input are allowed: The other five are reserved for record number and time.
TPCHK	Subroutine called to check for the tape ready condition, i.e. tape is on-line and not write protected.
ADCHK -	Subroutine called to check that the A/D convertor is not in mannual mode.
READØN -	Internal subroutine called to read the Systron Donner time code generator to determine when to start the test.
UNPACK -	Subroutine called to unpack the Systron Donner time from pseudo BCD to binary.
TEST -	Subroutine called to start the data acquisition and remain in control throughout the test.

4.3.3.1.4 Input Data - The following parameters must be available for storage/use by BEGIN

BUFR	A 16 word teltype input buffer filled by TYPIN.
NØRATE	Number of A/D rates
CLCNT	Number of microseconds between highest rate reads of the A/D convertor
MAXBUF	Maximum data buffer length (default 1000_{10})
BUFZER	Starting address minus 1 for data buffer zero. (Set at $77C_{16}$)
BUFZND	End address of buffer zero (default $B64_{16}$)
BUFØNE	Start address minus 1 for Buffer One (set to $B7D_{16}$)
NDBUF1	End address of buffer one (default $F65_{16}$)

HDBFLG	Header length on data buffer (set at 25)
MBUFLG	Minus buffer length including header (default=-1025_{10})
BGNCHN	First A/D channel to read (default = 0)
STHR STMIN STSEC STRMS	Systron Donner Strat hour, minute, second, and Millisecond (default is 0)
STPHR STPMIN STPSEL STØPMS	Systron Donner stop hour, minute, second, and millisecond (default is $1E00_{16}$)

In addition to the above values, the pointers to subroutines FETCH, CRLF, RESTØR, UNPACK, and TYPIN must be available in Scratch pad (locations 0 to 255). BUFR is stored at 256_{10} to 271_{16}. The binary numbers for the above values are returned from subroutine FETCH.

The inputs which may be alphabetical are put in the teletype buffer, BUFR, and then checked for the appropriate alphabetical value by BEGIN.

The hours, minute, second, and millisecond are input by internal subroutine READØN.

4.3.3.1.5 Output Data

The specific names in the input section may be changed if there is a default value. In addition to the above outputs, the following subroutines have calling arguments as indicated.

FETCH	-	The address of the message requesting numeric input is transfered through the x-register.
ØTL	-	The terminating character of a message is transfered through the S-register. The address pointing to the first word of the message is stored in the first location after the call to ØTL.
READØN	-	The code for the time to be returned (hours, minutes, seconds, milliseconds) is passed through the x register.
UNPACK	-	The pseudo BCD time code in passed to this subroutine through the A register.
CHKIT	-	The first character of the teletype buffer, BUFR, must be available. Since CHKIT is an internal subroutine within ± 255 of BUFR, no special action is required.

TYPIN_ Any single character to be output before filling the teletype buffer with input must be passed through the A register.

4.3.3.1.6. Storage Required - BEGIN requires 183_{16} or 387_{10} words of memory.

4.3.3.1.7 Description

The tasks of BEGIN are to: obtain all the required and optional inputs from the user; insure that the tape and A/D converter are ready; and to start the acquisition at the specified Systron Donner time (If the default values are used, the acquisition starts immediately). BEGIN Accomplish. The first task by calling the appropriate support subroutines to get a message to the user requesting input and return a binary number for those numeric inputs. In the case of a question, a yes or no response is required. If an error is detected in either the optional or mandatory input, the request for data is typed again. In the specific case of the number of microseconds per high speed read, BEGIN cannot determine if the error was legitimate. It requests that the user verify that the input is okay. The only correct input which can trigger the verification is if the number of microseconds is greater than 32767 and less than 65535.

The second task is accomplished by calling the A/D convertor and tape check routines. The A/D convertor checks only for the convertor being in manual mode. If so, the routine outputs diagnostic message and continues checking until the convertor is placed in automatic mode. The tape check routine checks for tape on-line and not write protected. If either/or both conditions are not true, a diagnostic is output. The condition(s) must be connected before BEGIN will continue.

After the tape and A/D convertor are checked, BEGIN will then start the test sequence by asking about writing a beginning end of file. The user can write an end of file to position past the first three inches of tape (recommended for high data rates). BEGIN will then request an S to begin checking the Systron Donner times against the start times. This allows the user a chance to review his input data before actually starting the test.

After the test acquisition is completed, the user is asked about continuing the acquisition with new times. This allows the user to have a short calibration run and then a long test without changing his fixed data. The tape can be rewound if the user desires. The GDAS then initializes itself and starts the whole sequence again.

4.3.3.2.0 Subroutine FETCH

4.3.3.2.1 The purpose of this subroutine is to output a message requesting numeric input and convert the subsequent input to a binary integer.

4.3.3.2.2 The calling sequence is LDX Address of message
JST FETCH
Returns here if input in error
Returns here if input is okay
Binary integer is in the A register.

4.3.3.2.3 Software/Software Interfaces.

FETCH	Interfaces with the following subroutines -
CRLF	Subroutine which issues a carriage return - line feed to the teletype
ØTL	Subroutine which writes a line to the teletype
TYPIN	Subroutine which fills the teletype input buffer.
CONVR	Subroutine which converts ASLII integers to a binary number.

4.3.3.2.4 Input Data - The address of the message to be output is transfered to FETCH through the x register. A binary number is returned from CØNVR in the A register, and if an error was detected, the overflow bit will be set. The address of the teletype buffer must be available in location $A4_{16}$ of scratchpad.

4.3.3.2.5 Output Data - Subroutine FETCH passes the terminating character of the message requesting input through the A register. The address of the first word of the message is stored in the first location after the call to the message writer, ØTL.

FETCH transfers a blank character to the teletype input routine, TYPIN, through the A register.

The address of the teletype buffer is transfered to the ASLII to binary conversion routine, CØNVR, through the x register.

The binary number that FETCH was requested to input is transfered to the calling program through the A register.

4.3.3.2.6 Storage Required - Subroutine FETCH requires D_{16} or 13_{10} locations.

4.3.3.2.7 Description

When FETCH is called, the address of the message requesting an input number is in the x register. FETCH stores that address in the first location after the call to the message writer, ØTL. FETCH then issues a carriage return-line feed by calling the routine, CRLF. After returning from CRLF, FETCH loads the terminating character into the A register before calling, ØTL. The terminating character is always a period (.) for any message from FETCH. After the message is printed by ØTL, FETCH calls the routine TYPIN to fill the teletype input buffer. Upon return from TYPIN, FETCH loads the address of the teletype buffer into the x register and calls the conversion routine, CONVR. When CONVR returns control to FETCH, a binary number will be in the A register. The overflow bit will be set (i.e., 1) if CONVR was unable to convert the number. If the overflow is set, FETCH returns control to the first location after the calling program. If the overflow is reset (i.e., 0) the return is to the second location after the call.

4.3.3.3.0 Subroutine CLØCK and STOPIT

4.3.3.3.1 The purpose of this subroutine is to respond to the real time clock interrupt and to start the data acquisition. The purpose STOPIT is to reset the clock subrout'ne.

4.3.3.3.2 There is no calling sequence since this is an interrupt subroutine.

4.3.3.3.3 Software/Software Interfaces.

CLOCK calls internal subroutine.
STRTAD to reset and start the A/D Convertor.
Internal subroutine STØPIT is called by external programs to reset the CLØCK routine variables to their initial state.

4.3.3.3.4 Input Data-There are no data inputs through calling sequences. However, the following variables must be available in scratchpad (locations $0\text{-}256_{10}$)

LOCATION	VARIABLE	DESCRIPTION
81_{16}	NØRATE	Number of Rates (1, 2, or 3)
82_{16}	NØLØ	Number of Low Speed A/D channels
83_{16}	LOCNT	Number of High speed reads per low speed read
84_{16}	NOMED	Number of medium speed channels.
85_{16}	MEDCNT	Number of high speed reads per medium speed read
86_{16}	NØHS	Number of high speed channels
87_{16}	BUFZER	Starting address for storing data into buffer zero.
88_{16}	BUFZND	Ending address for storing data into buffer zero.
89_{16}	BUFØNE	Address for buffer one.
$8A_{16}$	NDBUF1	
$8C_{16}$	MAXBUF	Data buffer length
$8D_{16}$	TPØUT	Tape output flag

LOCATION	VARIABLE	DESCRIPTION
$8E_{16}$	DONCLK	Systron Donner time conversion flag.
$8F_{16}$	BGNCHN	Beginning A/D channel number
91_{16}	DØNMS	Temporary storage for Systron Donner times (Hours, minutes, seconds, and milliseconds)
92_{16}	DNSEC	
93_{16}	DNMIN	
94_{16}	DNHRS	
$8B_{16}$	BUFUSE	Flag indicating which data buffer is currently being filled (buffer zero or one).

Subroutine STOPIT sets the following variables to their initial state.

BUFEND	End of Buffer sentinel set to 0.
DØNCLK	Systron Donner convert flag set to 0
CNTMED	Medium A/D rate internal counter to -1
CNTLO	Low A/D rate internal counter to -1
ADCNUM	Number of A/D chanels to read to -1

4.3.3.3.5 Output Data - CLOCK outputs the start codes to the A/D convertor and sets up the automatic input locations for the digital input values.

4.3.3.3.6 Storage Required - The CLOCK program requires 66_{16} or 102_{10} locations. This includes 14_{10} locations required by STØPIT.

4.3.3.3.7 Description - When a clock interrupt occurs, the CLOCK subroutine responds by clearing the interrupt. Since the time period between high speed A/D reads was used as a clock count, each interrupt signals the start time for high speed data acquisition. The low and medium speed counters are checked (and incremented) to determine if the low and medium speed channels are to be read on this particular interrupt. After the number of channels to be read is determined, CLOCK checks the buffer to insure that all the data to be converted will fit. If so, the acquisition is started. If the data will not fit, the bufferes are switched and the A/D convertor is started. The systron donner times are read and the donner time convert and tape output flags are set.

In either case, control is returned to the interrupted program.

When the test is complete, STOPIT is called by an external program. This subroutine stops the A/D convertor and the clock to prevent future conversions. STOPIT also resets counters interal to CLOCK.

4.3.3.4.0 Subroutine TEST

4.3.3.4.1 The purpose of this subroutine is to control the background tasks during a data acquisition.

4.3.3.4.2 The calling sequence is JST * TEST

4.3.3.4.3 Software/Software interfaces.

Subroutine TEST calls the following programs.

CLOCK	Called for the first data acquisition All additional acquisitions are done with clock interrupts.
TDUMP	Called to start the data transfer to magnetic tape and to write end of files on the tape. The A register contains the minus number of words to be transfered and the X register contains the buffer address minus one. The A register is loaded with -1 for writing end of files.
UNPACK	Called to convert the Systron Donner times from packed BCD to binary numbers. The A register contains the number to be converted.
STØPIT	Called to stop the data acquisition and reset the CLOCK routine.
OTL	Called to output messages to the teletype
ODEC	Called to convert a binary number to decimal and output it to the teletype
CRLF	Called to output a carriage return-line feed.

4.3.3.4.4 Input Data - There are no data passed as calling arguments. However, the following variables must be available in scratchpad (locations $0\text{-}255_{16}$)

LOCATION	VARIABLE	DESCRIPTION
$8B_{16}$	BUFUSE	Buffer in use flag.
$8E_{16}$	DØNCLK	Donner Clock time conversion flag.
80_{16}	CLCNT	Clock count (time between interrupts) in microseconds.

4.3.3.4.5 Output Data - The following messages are output from TEST.

1 - DONE - ERRØRS -- PARITY XXXX TIMING. XXXX

2 - XXXX RECØRDS WRITTEN.

3 - RATE TØØ FAST!

Message one and two give numeric values for the parity and timing errors and the number of records written.

Message three indicates that one buffer is full before the previous buffer has been completely output. This is an abort condition.

4.3.3.4.6 Storage required. TEST Requires AB_{16} or 171_{10} locations.

4.3.3.4.7 Description. When TEST is called, the appropriate counters (parity error, timing, tape output, etc.) are set to zero and the appropriate I/O devices are set up. The clock is started and subroutine CLOCK is called to start the data acquisition. TEST then enters a loop. The first step of the loop is to check for a buffer ready for output and tape not busy. If a buffer is ready, it is output with a call to TDUMP. If not, then the second step of the loop is executed. The second step checks for Systron Donner times to be converted. If there are times to be converted, then the routine UNPACK is called. After the times are unpacked and stored for output, the test stop times are checked to see if the test has exceeded the time. If so, the test is stopped. If not, the third step of the loop is executed. This step looks for an 'X' character from the teletype. If an X has been input, the test is stopped. If an X has not been input, the loop is repeated.

After the test has been completed, subroutine TEST writes two end of files on the tape and backspaces over the second. It then informs the user of any tape errors and of the total number of records written on tape. Control is then returned to the calling program.

$8D_{16}$	TPØUT	Tape output flag.
91_{16}	DONMS	Temporary storage for Systron Donner times.
92_{16}	DNSEC	
93_{16}	DNMIN	
94_{16}	DNHRS	
99_{16}	STPHR	Test stop times in hours, minutes, seconds, and milliseconds.
$9A_{16}$	STPMIN	
$9B_{16}$	STPSEC	
$9C_{16}$	STØPMS	
$9D_{16}$	BUFHDI	Address of the data buffers.
$9E_{16}$	BUFHDO	
$9F_{16}$	MBUFLG	Minus data buffer length.
$A1_{16}$	PAR	Tape parity error counter.
$A2_{16}$	TIMG	Tape timing error counter.
$A6_{16}$	TAPEND	Tape end of operation flag.

In addition, the addresses of all the external subroutines are stored in scratchpad.

LOCATION	ROUTINE
$3B_{16}$	CLØCK
71_{16}	CRLF
72_{16}	ØTL
73_{16}	UNPACK
75_{16}	TDUMP
76_{16}	ØDEC
$7B_{16}$	STØPIT

4.3.3.5.0 Subroutine RESTØR

4.3.3.5.1 The purpose of this subroutine is to reset all I/O devices and to restore all scratchpad variables to their initial state.

4.3.3.5.2 Calling Sequence. The calling sequence is JST*RESTØR

4.3.3.5.3 Software/Software Interfaces - none

4.3.3.5.4 Input Data - none

4.3.3.5.5 Output Data - The following list of scratchpad variables are restored to the indicated values.

LOCATION	VARIABLES	VALUE
80_{16}	CLCNT	0
81_{16}	NORATE	0
82_{16}	NØLØ	0
83_{16}	LØCNT	0
84_{16}	NØMED	0
85_{16}	MEDCNT	0
86_{16}	NØHS	0
87_{16}	BUFZER	$77C_{16}$
88_{16}	BUFZND	$B64_{16}$
89_{16}	BUFØNE	$B7D_{16}$
$8A_{16}$	NDBUF1	$F65_{16}$
$8C_{16}$	MAXBUF	1025
$8F_{16}$	BGNCHN	0
90_{16}	ADCRD	0
95_{16}	STHP	0
96_{16}	STMIN	0
97_{16}	STSEC	0

LOCATION	VARIABLE	VALUE
98_{16}	STRMS	0
99_{16}	STPHR	0
$9A_{16}$	STPMIN	0
$9B_{16}$	STPSEC	0
$9C_{16}$	STOPMS	0
$9D_{16}$	BUFHDI	$B64_{16}$
$9E_{16}$	BUFHDO	763_{16}
$9F_{16}$	MBUFLG	-1025
$A0_{16}$	MXBDUF	1000
$A3_{16}$	HDBFLG	25
$A6_{16}$	TAPEND	1

The twenty five header words in each data buffer are also filled with zeros.

4.3.3.5.6 Storage Required - This routine requires 41_{16} or 65_{10} locations.

4.3.3.5.7 Description - The I/O devices are reset first. Then the appropriate values are restored by using constants defined in RESTØR or by constants defined by load A register immediate instructions.

4.3.3.6.0 Subroutine EINS, ZWEI, and DREI

4.3.3.6.1 The purpose of these subroutines is to input Data generic to the number of A/D conversion rates.

4.3.3.6.2 Calling Sequence - The calling sequences are -

```
I         JST * DREI
I + 1     NØP
I + 2     NØP
I + 3     RETURNS HERE

I         JST * ZWEI
I + 1     RETURNS HERE

I         JST * EINS
I + 1     NØP
I + 2     RETURNS HERE
```

4.3.3.6.3 Software/Software Interfaces - Subroutine FETCH is called to type the data request and read the input value.

4.3.3.6.4 Input Data - The only data input is through the teletype and is returned in the A register by FETCH.

The following input values are required by the indicated entry point.

ENTRY POINTS	VARIABLES REQUIRED
EINS, ZWEI, DREI	NØHS number of High speed A/D channels.
ZWEI, DREI	NØMED number of medium speed a/D channels
DREI	NØLØ number of low speed A/D channels
ZWEI, DREI	MEDCNT number of times to read high speed channels before reading medium speed channels
DREI	LØCNT number of times to read high speed channels before reading low speed channels.

4.3.3.6.5 Output Data - The Data which are teletype inputs are stored in scratchpad (locations 0-255). In addition, descriptive messages requesting input are passed to FETCH to be output to the teletype.

4.3.3.6.6 Storage Required - The total storage required by the three entry points is 71_{16} or 113_{10} locations.

4.3.3.6.7 Description - Each subroutine only types out that information which is generic to its specific rate. The X register is loaded with the address of the message to be output. Subroutine FETCH is called to type out the message and read the input number. If the number is in error (i.e., couldn't be converted from ASCII), FETCH returns to the next instruction past the call. The message loop is then repeated If the value is valid, it is stored and the next call to FETCH is executed. All the entry points return control to the calling program.

4.3.3.7.0 Subroutine HEADER

4.3.3.7.1 The purpose of this subroutine is to accept user input data for the data buffer header words 6-25.

4.3.3.7.2 The Calling sequence is JST * HEADER

4.3.3.7.3 Software/Software Interfaces -
Subroutine HEADER calls: CRLF to output carriage return-line feeds; OTT to output a single character from the A register; ØTL to output a message; and FETCH to output a message and to read a number for storage.

4.3.3.7.4 Input Data - The following data must be available in scratchpad -

$9D_{16}$	BUFHDI	Address of the buffer header for
$9E_{16}$	BUFHDO	buffer one and zero.
$A4_{16}$	BUFTTY	Address of the teletype input buffer.

The only other input data is from the teletype for storage in the headers.

4.3.3.7.5 Output Data - up to 20 words of information may be stored in the buffer headers. Output message addresses are passed to FETCH and ØTL. A single character to be output is transferred to ØTT via the A register.

4.3.3.7.6 Storage Required - HEADER requires 57 or 87 locations.

4.3.3.7.7 Description - HEADER calls ØTL to output the message:

HEADER DATA INPUT TO TERMINATE, INPUT 'A/'. The subroutine FETCH is called to write the word number of the header in to which the next input value will go. If the value input is in error, the FETCH is called to print the message again. If the value is valid, it is stored in the headers. The header word pointers are incremented and FETCH is called again with the next word number. The process is repeated until either 20 values are input or the quantity "A/" is input. HEADER then returns control to the calling program.

4.3.3.8.0 Subroutine PFAIL

4.3.3.8.1 The purpose of the subroutine is to provide an orderly shutdown of the CF16A during a power failure.

4.3.3.8.2 There is no calling sequence since this is an interrupt subroutine.

4.3.3.8.3 Software/Software Interfaces
This subroutine doesn't call any other subroutines. However, it does store a jump to the restart routine at the interrupt location, 00_{16}.

4.3.3.8.4 Input Data - None

4.3.3.8.5 Output Data - An instruction is stored at the restart interrupt location

4.3.3.8.6 Storage Required - This routine requires 7_{16} or 7_{10} locations.

4.3.3.8.7 Description - When a power failure is detected, an interrupt is generated. This subroutine responds to that interrupt by halting an tape I/O. It also stores a jump to the restart routine at location 0. The interrupts are enable and the CFIGA is halted.

4.3.3.9.0 Subroutine PWRØN

4.3.3.9.1 The purpose of PWRØN is to insure an orderly restart after a power failure.

4.3.3.9.2 There is no calling sequence since this routine responds to an interrupt at location 0.

4.3.3.9.3 Software/Software Interfaces - This subroutine calls RESTØR to reset the system variables to their original state. It also calls STØPIT to reset the CLOCK data acquisition routine to its original state.

4.3.3.9.4 Input Data - None

4.3.3.9.5 Output Data - This subroutine stores a jump to the spurious interrupt handler at location 0.

4.3.3.9.6 Storage Required - This routine requires 9_{16} locations.

4.3.3.9.7 Description - When a restart interrupt occurrs at location 0, this routine calls RESTØR, STØPIT, and then restores the spurious interrupt subroutine call at location 0. (When a device requests an interrupt and does not respond with an address, the CF16A interrupts to location 0.) PWRØN then begins execution at location 111_{16}, the start of GDAS.

4.3.3.10.0 Subroutine BADRPT

4.3.3.10.1 The purpose of this subroutine is to alert the user that a spurious interrupt has occurred.

4.3.3.10.2 There is no calling sequence since this is an interrupt subroutine.

4.3.3.10.3 Software/Software Interfaces - This subroutine calls ØTL to output the message -

"SPURIOUS INTERRUPT! CYCLE TO CONTINUE."

4.3.3.10.4 Input Data - None

4.3.3.10.5 Output Data - The ASCII character '.' is loaded into the A register before calling ØTL.

4.3.3.10.6 Storage Required - This subroutine requires $1C_{16}$ or 27_{10} locations.

4.3.3.10.7 Description - When a device requests an interrupt and does not respond with an interrupt address, the CF16A interrupts to location 0. This subroutine issues a halt I/O to the tape controller and then outputs a message to alert the user of the interrupt and halts. Although the user can restart at the point of interruption, this interrupt usually signifies a hardware fault and is likely to recur until the hardware is repaired.

4.3.3.11.0 Subroutine TPCHK

4.3.3.11.1 The checks the tape unit to insure that a tape is mounted and not wirte protected.

4.3.3.11.2 Calling sequence - the calling sequence is JST * TPCHK.

4.3.3.11.3 Software/Software Interfaces - TPCHK calls CRLF to issue a carriage return-line feed to the teletype. Subroutine ØTL is called to output the message.

"TAPE NOT ON-LINE."

" TAPE IS WRITE PROTECTED."

4.3.3.11.4 Input Data - None

4.3.3.11.5 Output Data - The message termination character is transfered to ØTL via the A register. The address of the message to be output is transfered as an argument immediately after the call.

```
LDA termination Character
JST * ØTL
DATA message address.
```

4.3.3.11.6 Storage Required - This subroutine requires 30_{16} or 48_{10} locations.

4.3.3.11.7 Description - Using the sense instruction, TPCHK checks to insure that the tape is on-line and not write protected. If either of these conditions are not met, a message is output. TPCHK then enters a loop until the error condition is corrected. After the error conditions are corrected, TPCHK returns to the calling program.

4.3.3.12.0 Subroutine ADCHK

4.3.3.12.1 Subroutine ADCHK checks the A/D convertor for automatic mode.

4.3.3.12.2 Calling Sequence - The calling sequence is
JST *ADCHK

4.3.3.12.3 Software/Software Interfaces - This subroutine calls CRLF to output a carriage return-line feed and ØTL to output the message

"ADC IN MANUAL MØDE."

4.3.3.12.4 Input Data - None

4.3.3.12.5 Output Data - The termination character of the message is transferred to ØTL via the A register. The address of the message is transferred as a calling argument.

```
LDA Termination Character
JST * ØTL
DATA Message Address
```

4.3.3.12.6 Storage Required - ADCHK required 16_{16} or 22_{10} locations.

4.3.3.12.7 Description - Using the sense instruction, ADCHK determines whether or not the A/D convertor is in manual mode. If so, ADCHK outputs a message to the user and waits for the A/D converter to be put in automatic mode. ADCHK returns to the calling program.